DESIGNING HIGH PERFORMANCE LOWER COST TABLETS WITH NEWEST TECHNOLOGIES USING OPEN SOURCE OPERATION SYSTEM

YODCHAI SINGHSATHITSUKH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF SCIENCE
(INFORMATION TECHNOLOGY MANAGEMENT)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2015

Thesis
entitled

DESIGNING HIGH PERFORMANCE LOWER COST TABLETS WITH NEWEST TECHNOLOGIES USING OPEN SOURCE OPERATION SYSTEM

…….....
Mr. Yodchai Singhsathitsukh
Candidate

.....
Asst. Prof Supaporn Kiattisin,
Ph.D. (Electrical and Computer Engineering)
Major advisor

.....
Asst. Prof Adisorn Leelasantitham,
Ph.D. (Electrical Engineering)
Co-advisor

.....
Lect. Taweesak Samanchuen,
Ph.D. (Electrical Engineering)
Co-advisor

…….....
Prof. Patcharee Lertrit,
M.D., Ph.D. (Biochemistry)
Dean
Faculty of Graduate Studies,
Mahidol University

…….
Asst. Prof Supaporn Kiattisin,
Ph.D. (Electrical and Computer Engineering)
Program Director
Master of Science Program in Information Technology Management
Faculty of Engineering
Mahidol University

Thesis
entitled

DESIGNING HIGH PERFORMANCE LOWER COST TABLETS WITH NEWEST TECHNOLOGIES USING OPEN SOURCE OPERATION SYSTEM

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Science
(Information Technology Management)
on
October 16, 2015

…….....
Mr. Yodchai Singhsathitsukh
Candidate

.....
Asst.Prof. Santipat Arunthari
Ph.D. (Information Systems)
Chair

…….....
Asst. Prof Supaporn Kiattisin,
Ph.D. (Electrical and Computer Engineering)
Member

…….
Asst. Prof Adisorn Leelasantitham,
Ph.D. (Electrical and Computer Engineering)
Member

…….....
Lect. Theera Piroonratana,
Ph.D. (Electrical Engineering)
Member

…….
Lect. Taweesak Samanchuen,
Ph.D. (Electrical Engineering)
Member

…….....
Prof. Patcharee Lertrit,
M.D., Ph.D. (Biochemistry)
Dean
Faculty of Graduate Studies,
Mahidol University

…….
Asst. Prof. Jackrit Suthakorn,
Ph.D. (Robotics)
Dean
Faculty of Engineering
Mahidol University

ACKNOWLEDGEMENTS

The writer would like to thank Meme and Rossarin, the writer's heart and soul. Your love and support has sustained me throughout the journey, which was very hard as I was a small fish in such a large IT market with vast competition. Furthermore, working with Chinese counterparts was difficult due to language barriers. However, with your support, I achieved my goal of creating the first tablet prototype which will always be remembered as one of the largest achievement milestones in my life. Despite the fact that the prototype was short-lived, I took the plunge and gave it a shot. The researcher (experienced in CPU Core Architecture and IRQ Placement and Firmware Development) worked on the project with Mr. Brian Liu (experienced in open source software development) and Mr. Sawyer (experienced in electronic circuit). Special thanks to Mr. Brian and Mr. Sawyer of Shenzhen Seedmorn Technology Co., Nanshan Software Park; Nanshan District; Shenzhen, China. http://www.seedmorn.cn/.

Thank you Asst. Prof Supaporn Kiattisin, Ph.D and Asst. Prof.Adisorn Leelasantitham for your great advice and encouragement. I may be one of the most difficult students but with your patience and time, I made it. No words can explain the sacrifice and support that you have given me these past years. Thank you once again

Yodchai Singhsathitsukh

DESIGNING HIGH PERFORMANCE LOWER COST TABLETS WITH NEWEST TECHNOLOGIES USING OPEN SOURCE OPERATION SYSTEM

YODCHAI SINGHSATHITSUKH 5738598 EGIT/M

M.Sc. (INFORMATION TECHNOLOGY MANAGEMENT)

THESIS ADVISORY COMMITTEE: SUPAPORN KIATTISIN, PH.D., ADISORN LEELASANTITHAM, PH.D., TAWEESAK SAMANCHUEN, PH.D.

ABSTRACT

This research study about the effectiveness of I/O IRQ, firmware and drivers which have an effect on low-cost tablets. The researcher gather the data from related research. Then the processor was designed and the I/O mechanism was developed. While doing the experiments, statistical analysis and the chi-square testing for the measuring the performance of tablets was undertaken. The experiment concluded that designed I/O IRQ and firmware drivers would make developed tablets have a higher efficiency and a lower-cost than tablets which are manufactured from imported I/O IRQs (sold in local markets) with similar features.

KEY WORDS: IRQ / I/O / FIRMWARE DRIVERS / LOW COST TABLET / DMA

58 pages

การออกแบบคอมพิวเตอร์แทบเล็ทต้นทุนต่ำ โดยใช้ระบบปฏิบัติการแบบเปิดเผยซอร์สโค้ด
DESIGNING HIGH PERFORMANCE LOWER COST TABLETS WITH NEWEST TECHNOLOGIES USING OPEN SOURCE OPERATION SYSTEM

ยอดชาย สิงห์สถิตสุข 5738598 EGIT/M

วท.ม. (การจัดการเทคโนโลยีสารสนเทศ)

คณะกรรมการที่ปรึกษาวิทยานิพนธ์ : สุภาภรณ์ เกียรติสิน, Ph.D., อดิศร ลีลาสันติธรรม, Ph.D., ทวีศักดิ์ สมานชื่น, Ph.D.

บทคัดย่อ

งานวิจัยนี้ศึกษา ผลกระทบของ I/O IRQ และไดรเวอร์เฟิร์มแวร์ต่อประสิทธิภาพคอมพิวเตอร์แทบเล็ตต้นทุนต่ำ ข้อมูลที่ใช้ในการศึกษารวบรวมจากงานวิจัยที่เกี่ยวข้อง และทำการออกแบบหน่วยประมวลผลและพัฒนนากลไก I/O ของตนเอง พร้อมนำทดสอบการใช้งานและวิเคราะห์ทางสถิติ การทดสอบด้วยไดสแควร์ เพื่อหาประสิทธิภาพการทำงานของคอมพิวเตอร์แทบเล็ต จากการทดลองสรุปได้ว่าการออกแบบ I/O IRQ และไดรเวอร์เฟิร์มแวร์ของตนเองทำให้คอมพิวเตอร์แทบเล็ตที่พัฒนาขึ้นมีประสิทธิภาพสูงและต้นทุนต่ำกว่าคอมพิวเตอร์แทบเล็ตที่ผลิตขึ้นจากการนำเข้า I/O IRQ ที่มีขายในท้องตลอดและมีคุณลักษณะของคอมพิวเตอร์แทบเล็ตที่ใกล้เคียงกัน

58 หน้า

CONTENTS

	Page
ACKNOWLEDGEMENTS	**iii**
ABSTRACT (ENGLISH)	**iv**
ABSTRACT (THAI)	**v**
LIST OF TABLES	**viii**
LIST OF FIGURES	**ix**
CHAPTER I INTRODUCTION	**1**
1.1 Background	1
1.2 Objectives of study	2
1.3 Scope of work	2
1.4 Expected result	3
CHAPTER II LITERATURE REVIEW	**4**
2.1 I/O	4
2.2 IRQ	11
2.3 Firmware Drivers	13
2.4 Relationship between I/O, IRQ, and Firmware Drivers	14
2.5 Lower Cost Tablets	16
2.6 Related Studies	19
2.7 Theoretical, Conceptual, and Operational Frameworks	24
CHAPTER III RESEARCH METHODOLOGY	**27**
3.1 Variables and Measurement	27
3.2 Research Design and Analysis	31
CHAPTER IV RESULTS AND DISCUSSION	**32**
4.1 I/O	32
4.2 Interrupt	33
4.3 Firmware Driver	34
4.4 Dummy Variable	34

CONTENTS(cont.)

	Page
4.5 Variables Combined	34
4.6 Efficiency Test	36
4.7 Comparison through Chi Square Test	37
4.8 Discussion	38
CHAPTER V CONCLUSION	**51**
5.1 Benefits of Study	51
5.2 Problems of Study	52
5.1 Limitations of Study	53
5.2 Future Recommendations	54
REFERENCES	**56**
BIOGRAPHY	**58**

LIST OF TABLES

Table		Page
2.1	I/O Devices and Data Rates.	4
2.2	Characteristics of I/O Devices.	9
3.1	I/O Speeds.	27
3.2	Maskable Interrupts.	28
3.3	CPU Research and Scores.	30
4.1	I/O Variable (A).	32
4.2	IRQ Variable (B).	33
4.3	Dummy Variable (N).	34
4.4	Variables A, B, C, and N.	35
4.5	Results of Efficiency Calculations.	36
4.6	Chi Square Table.	37

LIST OF FIGURES

Figure		Page
2.1	I/O Architecture.	5
2.2	Kernel I/O Structure.	7
2.3	Lifecycle of I/O Request.	8
2.4	Basic Operations of Interrupt.	12
2.5	Conceptual Framework.	25
2.6	Operational Framework.	26

CHAPTER I
INTRODUCTION

The goal of this chapter is to describe the reasons for undertaking the research study, how the research problem can be resolved, the objective and scope of the study, and what the anticipated benefits of the study will be.

1.1 Background

The iPad is a popular device used by millions around the world. For instance, since the third quarter of 2010, the top five quarterly sales were:

1) 26.04 million iPad sales in the first quarter of 2014;
2) 22.86 million iPad sales in the first quarter of 2013;
3) 21.42 million iPad sales in the first quarter of 2015;
4) 19.48 million iPad sales in the second quarter of 2013; and
5) 17.04 million iPad sales in the third quarter of 2014 [1].

As a result, it can be concluded that other types of tablets have increased in popularity. However, most companies, Apple included, are focused on making a profit. Therefore, a primary goal of these companies has been to integrate new technological designs in an effort to create a high performing, yet low cost tablet.

Knowing this, there are numerous types of tablets available for purchase. In 2015, for example, the ten best tablets available for purchase include the iPad Air 2, Nexus 7, iPad Air, Nexus 9, iPad mini 2, Sony Xperia Z2 Tablet, Bush MyTablet 8, Samsung Galaxy Tab S 10.5, Sony Xperia Tablet Compact, and Samsung Galaxy Tab S 8.4 [2]. Significantly, Apple’s iPad has three of the top ten tablets available for purchase, while Samsung has two of the top ten tablets available for purchase.

One of the reasons for this is that both companies design their own CPUs. In fact, in early 2015, “business Korea report that Samsung's large scale integration

(LSI) division recently began developing a custom CPU core project for mobile application processors" [3]. This has been seen as being one of the most important steps taken by the company in order to "establish Samsung Semiconductor as a maker and supplier of grade-a silicon for mobile device vendors" [3].

On the one hand, this is an unsurprising move for the company. On the other hand, this is incredible, especially considering that Apple hired one of Samsung's top engineers and designers, Jim Mergard, in 2012. This was a strong move by Apple because "it signifies Apple's commitment to designing its own chips. Its A6 chip in the iPhone 5 was its first chip to use its own custom core. While Apple had been using chips customized for its products (the A4, A5, A5X, etc.), those were built on ARM's architecture—the Cortex-A8 and A9. In simple terms, Apple had been just tweaking things here and there with the blueprints, but with the A6, it built the chip design from the ground up using just ARM's reference designs, not its ready-made cores. That is a very big deal in the world of chip design, and it is what let Apple boost the A6's performance and battery life at once" [4].

Since these two companies design their own chips, they are in control of where the I/Os are placed on the circuit board, which allows them to create optimal efficiency and output. As a result, interrupt requests specification and selections from the drivers in the firmware is important for CPUs to specify the I/O correctly to obtain optimized output from the I/Os.

1.2 Objectives of study

1.2.1 Explore how IRQs and firmware drivers make I/Os work better for cheaper tablets.

1.3 Scope of work

1.3.1 CPU design by Samsung, Apple, and other vendors.

1.3.2 IRQ and firmware drivers by Samsun, Apple, and other vendors.

1.3.3 Maximum output from I/Os.

1.4 Expected result

1.4.1 Through knowing the exact location of I/O placement on the PC board for maximum output, Apple and Samsung are able to offer cheaper tablets that work better than other tablets that do not have customized I/Os

CHAPTER II
LITERATURE REVIEW

The goal of this chapter is to discuss I/O, IRQ, firmware drivers, and the relationships of these components. With this knowledge, it is possible to see how a cheaper tablet can be made. The chapter concludes with a comparison of similar studies, providing a framework for this specific study.

2.1 I/O

I/O has different goals. However, computers must have I/O in order to be useful. Despite this, I/O speeds vary. The following table shows examples of I/O devices and data rates:

Table 2.1 I/O Devices and Data Rates.

Device	Data Rate
Keyboard	10 bytes / sec
Mouse	100 bytes / sec
56K Modem	7 KB / sec
802.11g Wireless	6.75 MB / sec
52x CD-ROM	7.8 MB / sec
Fast Ethernet	12.5 MB / sec
Compact Flash Card	40 MB / sec
FireWire (IEEE 1394)	50 MB / sec
USB 2.0	60 MB / sec
SONET OC-12 Network	78 MB / sec
SCSI Ultra 2 Disk	80 MB / sec
Gigabit Ethernet	125 MB / sec
SATA Disk Drive	300 MB / sec
Ultrium Tape	320 MB / sec
PCI Bus	528 MB / sec

Operational parameters include: byte/block, sequential/random, and polling/interrupts. For instance, keyboards “provide a single byte at a time, while tapes or disks provide whole blocks. Tapes must be accessed sequentially, while disks or CDs can be accessed randomly. Some devices require continual monitoring, while others generate interrupts when they need service” [5]. Thus, the goal of the I/O subsystem is to “provide uniform interface for wide range of devices” [5]. The following code is one commonly used for many different devices:

```
int fd = open(“/dev/something”);
for (int i = 0; i < 10; i++) {
    fprintf(fd,” Count %d\n”,i);
}
close(fd);
```

This code is used to create a standard interface [5]. The I/O architecture is as follows:

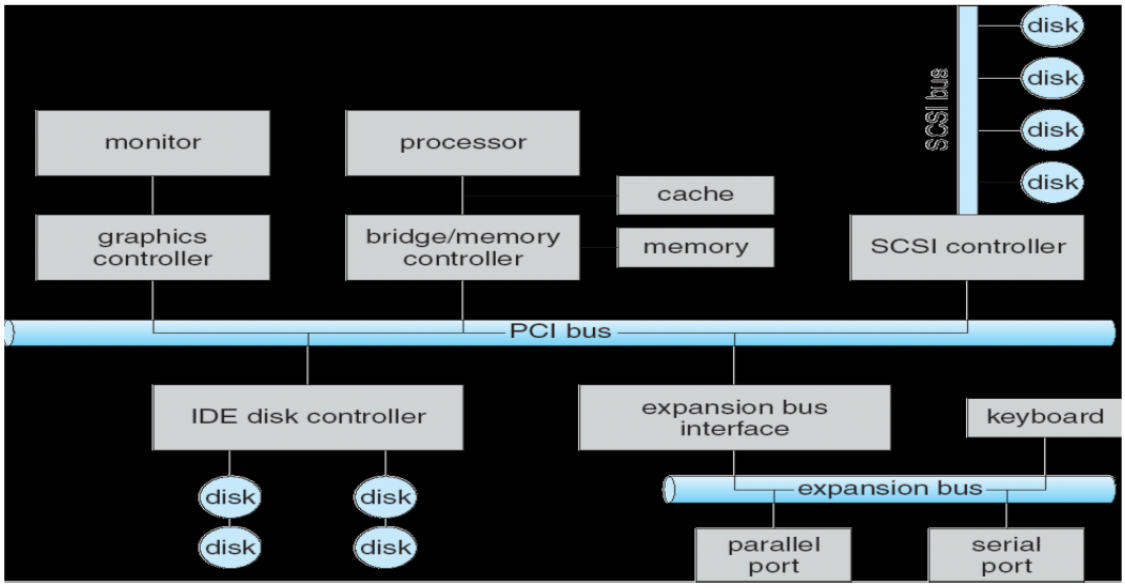

Figure 2.1 I/O Architecture.

The goal of this figure is to provide information regarding the general layout of the I/O architecture. Thus, it shows the relationship between the monitor, processor, graphics controller, bridge/memory controller, cache, memory, disk, SCSI controller, IDE disk controller, PCI bus, expansion bus interface, keyboard, expansion bus, parallel port, and serial port.

Research shows that the “CPU interacts with the device controller, which contains read/write control registers and may contain memory for request queues or bit-mapped images” [5]. Information is accessed in two different ways by the processor: 1) I/O instructions and/or 2) memory mapped I/O. I/O instructions contains “explicit in/out instructions” [5]. Memory mapped I/O loads and stores information. In this situation, “registers/memory appear in physical address space and I/O is accomplished with load and store instructions” [5]. However, there are differences found in memory mapped and explicit I/O. For instance, explicit I/O instructions require the use of assembly language. Furthermore, explicit I/O instructions “prevents user-mode I/O” [5]. Memory mapped I/O does not require special instructions and can be used in C. Furthermore, memory mapped I/O “allows user-based I/O” [5]. Finally, memory mapped I/O requires that caching addresses be prevented [5].

The application I/O interface varies. For instance, the I/O system “calls encapsulate device behaviors in generic classes; device – driver layer hides differences among I/O controllers from kernel; and devices vary in many dimensions, such as character-stream or block, sequential or random-access, sharable or dedicated, speed of operation, and read-write, read only, or write only” [5]. Within the kernel, the device-specific code “interacts directly with the device hardware, which supports a standard internal interface; allows the same kernel I/O system to interact easily with different device drivers; has special device-specific configuration that is supported with the ioctl() system call” [5]. Thus, the following figure shows a kernel I/O structure:

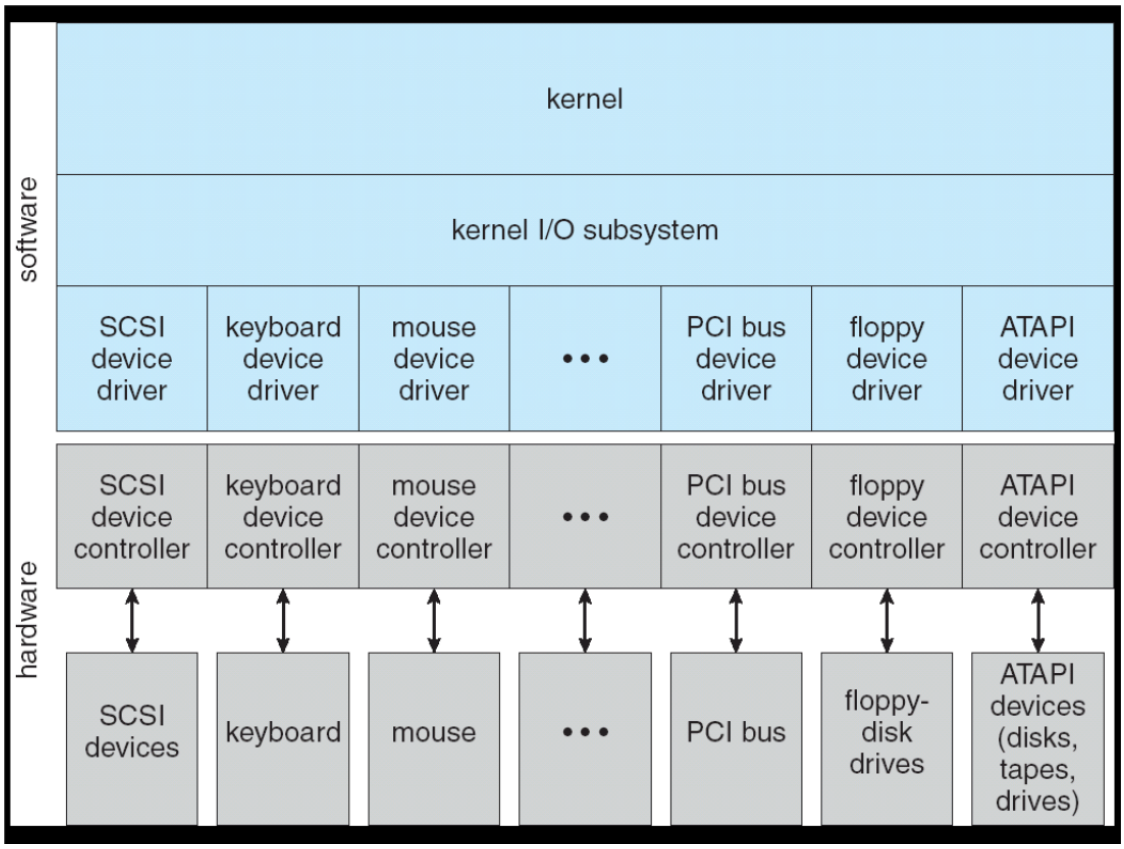

Figure 2.2 Kernel I/O Structure.

The goal of this figure is to display how the I/O structure of a kernel is developed and how it affects different devices. It shows the relationship between the kernel, the kernel I/O subsystem, SCSI device driver, keyboard device driver, mouse device driver, PCI bus device driver, floppy device driver, ATAPI device driver, SCSI device controller, keyboard device controller, mouse device controller, PCI bus device controller, floppy device controller, ATAPI device controller, SCSI devices, keyboard, mouse, PCI bus, floppy disk drives, and ATAPI devices.

The kernel is important because it “keeps state for I/O components, including open file tables, network connections, and character device state” [5].

There are typically two portions of device drivers: the top half and the bottom half. The top half is “accessed in the call path from system calls” [5]. The top half serves as “the kernel’s interface to the device driver and will start I/O to device, but may put thread to sleep until finished” [5]. Ultimately, the top half of the device

driver “implements a set of standard, cross-device cells like open(), close(), read(), write(), and ioctl(); and implements special VMAs to support mmap() based I/O” [5]. The bottom half “runs as an interrupt routine; gets input or transfers next block of output; may wake sleeping threads if I/O now complete; and may use deferred processing” [5]. The following figure shows the lifecycle of an I/O request:

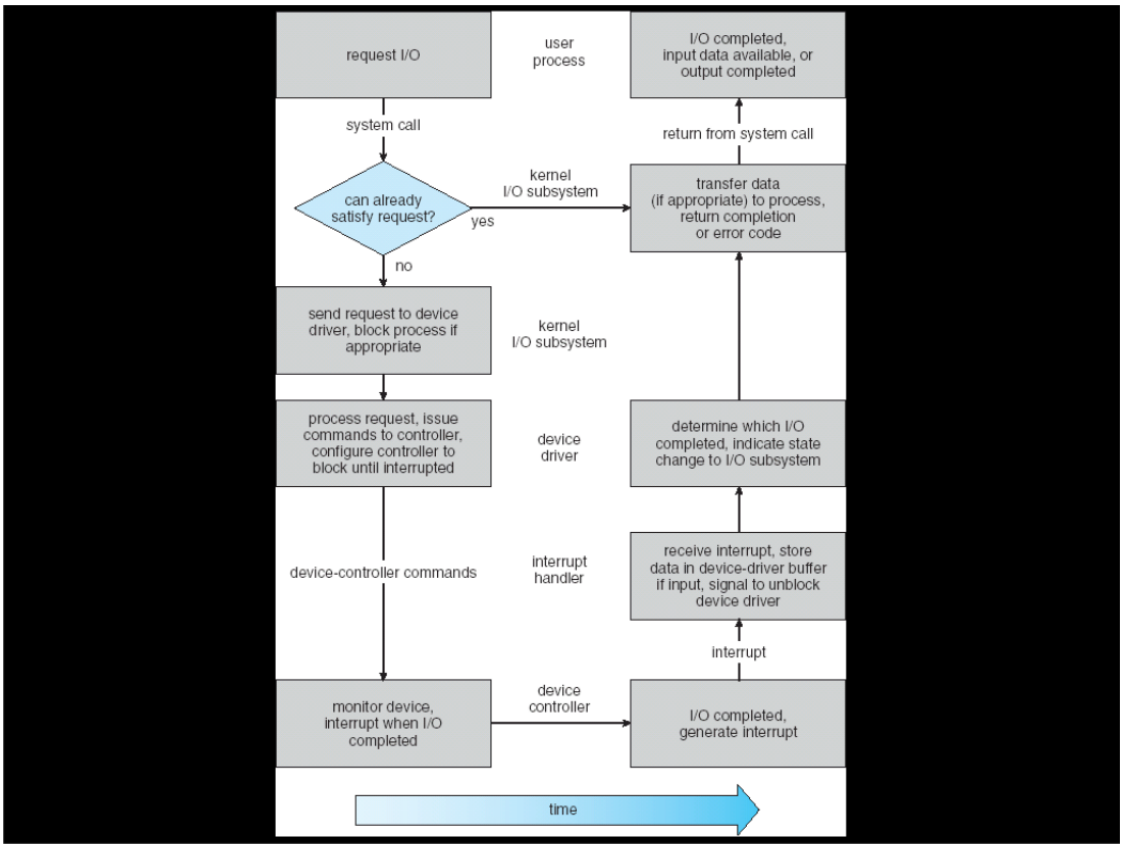

Figure 2.3 Lifecycle of I/O Request.

The purpose of this figure is to show how long it takes for an I/O request to be executed from start to finish. As noted in earlier research, there are numerous different characteristics of I/O devices. The following table shows these characteristics:

Table 2.2 Characteristics of I/O Devices.

Aspect	Variation	Example
Data-transfer mode	Character Block	Terminal Disk
Access method	Sequential Random	Modem CD-ROM
Transfer schedule	Synchronous Asynchronous	Tape Keyboard
Sharing	Dedicated Sharable	Tape Keyboard
Device speed	Latency Seek time Transfer rate Delay between operations	
I/O direction	Read only Write only Read-write	CD-ROM Graphics controller Disk

The goal of this table is to provide information regarding the differences in aspect and variation of particular I/O devices.

The different devices are block and character devices, network devices, clock devices and timers, and clock hardware. Block devices has commands, such as “read, write, or seek and has raw I/O or file-system access and memory mapped file access is possible” [5]. Character devices has commands, such as “get and put, inputs one character at a time, and libraries layered on top allow line editing” [5]. Network devices have unique interfaces, such as sock interface. Sock interface is unique because it “separates network protocol from network operation and includes select functionality” [5]. However, the approaches used with network devices vary and can include “pipes, FIFOs, streams, queues, or mailboxes” [5]. The purpose of clock devices and timers includes “maintaining the time of day, accounting for CPU usage, preventing processes from taking longer than they are allowed to, handling alarm system call made by user processes, providing watchdog timers for parts of the system itself, and doing profiling, monitoring, and statistics gathering” [5].

OS mechanisms involve “transferring data, notification, and buffering” [5]. When using a programmed I/O, data is transferred to/from the controller by transferring

“each byte via processor in/out or load/store” [5]. This is considered to be beneficial because it is a simple and easy to program hardware. However, it “consumes processor cycles proportional to data size” [5]. When using direct memory access, data is transferred to/from the controller by “giving the controller access to the memory bus and asking it to transfer data to/from memory directly” [5]. Direct memory access is beneficial because it allows for large data movement. However, it ‘requires a DMA controller and bypasses the CPU to transfer data directly between the I/O device and memory” [5].

Polling is important because it tells the OS when “the I/O device has completed an operation or the I/O operation has encountered an error” [5]. Polling is done when the “OS periodically checks a device-specific status register or the I/O device puts completion information in the status register” [5]. Although it is simple and has a “potentially low overhead,” polling includes a “busy-wait cycle to wait for I/O from the device, which may waste many cycles on polling if there are infrequent, expensive, or unpredictable I/O operations” [5]. Another option is an I/O interrupt. The I/O interrupt occurs “whenever the I/O device needs service” [5]. As a result, interrupts are beneficial when dealing with unpredictable events, yet typically involves a high overhead. However, some devices choose to utilize both polling and interrupts, such as the Intel E1000 Gigabit Ethernet Adapter, which uses an “interrupt for the first incoming packet and poll for subsequent packets until the hardware packet arrival queue is empty” [5].

As a result, it is found that I/O is “a major factor in system performance due to the demands of the CPU to execute device driver, kernel I/O code; context switches due to interrupts; data copying; and network traffic” [5]. System performance can be improved through “direct memory access; reduced/eliminated data copying; reduced number of context switches; and reduced interrupts by using large transfers, smart controllers, or polling” [5]. Other options involve utilizing blocking and non-blocking I/O. A blocking interface causes the process to wait by “putting the process to sleep until the data is ready for read or written for write” [5]. A non-blocking interface causes the process to not wait by “returning quickly from read or write request with count of bytes successfully transferred, where read may return nothing and write may write nothing” [5]. An asynchronous interface causes to process to notify the user later by

“when request data, take pointer to user’s buffer, return immediately; and later kernel fills buffer and notifies user; and when send data, take pointer to user’s buffer, return immediately; and later kernel takes data and notifies user” [5]. The kernel I/O subsystem has interfaces designed for device reservation, caching, scheduling, and spooling. Device reservation provides “exclusive access to a device” [5]. Caching provides a “fast memory holding copy of the data” [5]. Scheduling provides “I/O request reordering” [5]. Spooling provides the opportunity to “hold a copy of output for a device” [5]. Buffering is crucial because it “stores data in memory while transferring between devices [in order] to cope with device speed mismatch, to cope with device transfer size mismatch, or to maintain ‘copy semantics’” [5]. Thus, although buffering is important to performance, it can reduce performance as well. For instance, I/O requests to hardware operations can be done by “reading a file from disk for a process, which is done when the CPU can determine the device holding file, translate name to device representation, physically read data from disk to buffer, make data available to requesting process, and return control to process” [5].

Finally, there are interactions between I/O and virtual memory. For instance, “the kernel deals with (kernel) virtual addresses, which do not necessarily correspond to physical addresses” [5]. As a result, “contiguous virtual addresses are probably not contiguous in physical memory” [5]. This means that although “some systems have an I/O map – the I/O bus manager has a version of the virtual memory map, but more often, the kernel has to translate the virtual address itself” [5].

2.2 IRQ

Interrupts are important for the CPU to complete its processes effectively. In fact, “the CPU issues commands to the I/O module then proceeds with its normal work until interrupted by the I/O device on completion of its work” [6]. However, interrupts occur differently for input and output. Thus, “for input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The action actions to perform depend on whether the device uses I/O ports or memory mapping” [6]. For output, in contrast, “the device delivers an interrupt

either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer" [6]. The goal of interrupt is to prevent the CPU from "having to wait for the devices" [6]. The following figure shows the basic operations of interrupt:

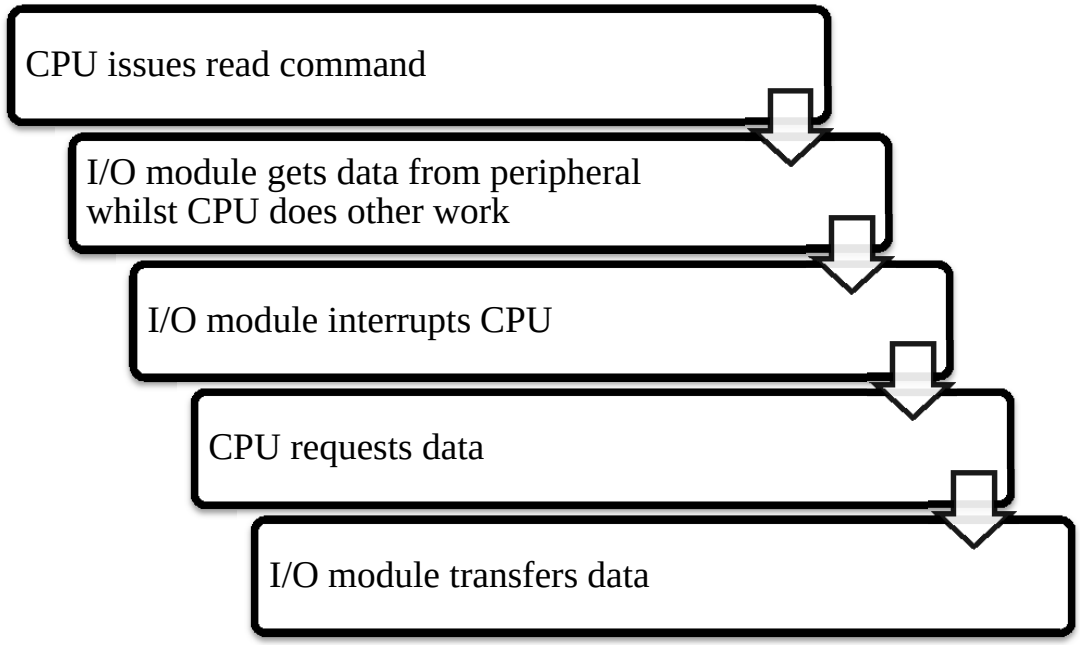

Figure 2.4 Basic Operations of Interrupt.

The goal of this figure is to show the basic operations that interrupt encompasses. This allows the reader to see how I/O is influenced through interrupts and how interrupts influences I/O. The steps include: CPU issues read command; I/O module gets data from peripheral whilst CPU does other work; I/O module interrupts CPU; CPU requests data; and I/O module transfers data.

As stated in the preceding section, there are numerous problems with polling, which results in the use of interrupts. In contrast to polling, interrupts are used so that "the running program is not responsible for checking the status of I/O devices. Instead, it simply does its own work and assumes that the I/O will take care of itself" [7]. In this scenario, the I/O is handled through an I/O subprogram, known as an interrupt service routine (ISR). In fact, "an interrupt can occur during any instruction cycle as long as interrupts are enabled. When the current instruction completes, the CPU interrupts the flow of the program, executes the ISR, and then resumes the

program. The program itself is not involved and is, in fact, unaware that it has been interrupted" [7]. Significantly, "interrupts can be globally enabled or disabled via the IEN flag and some architectures have a separate ISR for each device" [7]. Interrupts are advantageous because "user program progress is only halted during actual transfer". However, interrupts are disadvantageous because they require special hardware in order to "because an interrupt, detect an interrupt, or save the proper states to resume after the interrupts". Although similar, there are differences between interrupts and exceptions. For instance, "an I/O interrupt is asynchronous; is not associated with any instruction; does not prevent any instruction from completion; needs to convey the identity of the device generating the interrupt; and needs to be prioritized". Thus, through interrupts, the "processor issues the command and the device proceeds and leaves the processor free. Overcomes CPU waiting, avoids repeated checking of device by CPU through polling, and I/O module interrupts when ready".

2.3 Firmware Drivers

Firmware drivers have an impact on I/O speed. As found in preceding sections, I/O requests "are handled asynchronously by the operating system. This means that when a read or write request is made, the thread making it waits for the I/O to complete" [8]. As a result, the CPU is freed "for use by another thread" [8]. When the I/O request takes longer than anticipated, it can be caused by "I/O subsystem problems or misconfiguration, excessive I/O requests, data files not optimally placed on the disk, and fragmentation" [8].

Sub-optimal I/O performance is commonly caused by hardware errors. Two of the most common causes of "poor I/O throughput are out of date firmware and insufficient queue length" [8]. Another cause can be due to a file system filter driver. This is because these drivers are designed to "intercept requests before they reach the file system, and performs additional processing such as anti-virus checking and encryption" [8].

2.4 Relationship between I/O, IRQ, and Firmware Drivers

There is a distinct relationship between I/O, IRQ, and firmware drivers as evidenced in the preceding sections. The goal of this section is to tie this information together to explain the exact relationship between the individual components.

There are different goals within I/O. However, I/O must occur in order for the CPU to be useful. In contrast, the variance in I/O goals results in variances in I/O speeds. In section 2.1, it was noted that there were different data rates in I/O devices ranging from 10 bytes per second (keyboard) to 528 MB per second (PCI bus) [5], [9]. The same section also showed that the operational parameters of I/O include: byte/block, sequential/random, and polling/interrupts. The parameters varied dependent upon the type of device. For example, keyboards "provide a single byte at a time, while tapes or disks provide whole blocks. Tapes must be accessed sequentially, while disks or CDs can be accessed randomly. Some devices require continual monitoring, while others generate interrupts when they need service" [5]. Since the I/O needs to meet a variety of goals, the subsystem needs to provide a uniform interface that can meet the needs of multiple devices [5].

Information can be accessed in two different ways by the processor. The first way is through I/O instructions, which contains "explicit in/out instructions" [5]. The second way is through memory mapped I/O, which contains "load and store instructions" [5]. Therefore, there are differences in these methods. For example, explicit I/O instructions require the use of assembly language [5]. Memory mapped I/O does not require special instructions and can be used in C [5].

Device drivers are composed of two portions: top and bottom. The top half of the device driver tells the bottom half what to do [5]. Therefore, both portions work together in order to produce output. The top half serves as "the kernel's interface to the device driver and will start I/O to device, but may put thread to sleep until finished" [5]. Ultimately, the top half of the device driver "implements a set of standard, cross-device cells like open(), close(), read(), write(), and ioctl(); and implements special VMAs to support mmap() based I/O" [5]. The bottom half "runs as an interrupt routine; gets input or transfers next block of output; may wake sleeping threads if I/O now complete; and may use deferred processing" [5].

The different mechanisms used by CPUs include “transferring data, notification, and buffering” [5]. When using a programmed I/O, data is transferred to/from the controller by transferring “each byte via processor in/out or load/store” [5]. This is considered to be beneficial because it is a simple and easy to program hardware. However, it “consumes processor cycles proportional to data size” [5]. When using direct memory access, data is transferred to/from the controller by “giving the controller access to the memory bus and asking it to transfer data to/from memory directly” [5]. Direct memory access is beneficial because it allows for large data movement. However, it ‘requires a DMA controller and bypasses the CPU to transfer data directly between the I/O device and memory” [5].

Polling is important because it tells the OS when “the I/O device has completed an operation [and/or] the I/O operation has encountered an error” [5]. Polling is done when the “OS periodically checks a device-specific status register [or the] I/O device puts completion information in the status register” [5]. Although it is simple and has a “potentially low overhead,” polling includes a “busy-wait cycle to wait for I/O from the device, which may waste many cycles on polling if there are infrequent, expensive, or unpredictable I/O operations” [5]. Another option is an I/O interrupt. The I/O interrupt occurs “whenever the I/O device needs service” [5]. As a result, interrupts are beneficial when dealing with unpredictable events, yet typically involves a high overhead. However, some devices choose to utilize both polling and interrupts, such as the Intel E1000 Gigabit Ethernet Adapter, which uses an “interrupt for the first incoming packet and poll for subsequent packets until the hardware packet arrival queue is empty” [5]. This is confirmed in section 2.2 with the discussion of how interrupts allow the CPU to complete its processes effectively. In fact, “the CPU issues commands to the I/O module then proceeds with its normal work until interrupted by the I/O device on completion of its work” [6]. Interrupts are advantageous because “user program progress is only halted during actual transfer”. However, interrupts are disadvantageous because they require special hardware in order to “because an interrupt, detect an interrupt, or save the proper states to resume after the interrupts”.

As a result, it is found that I/O is “a major factor in system performance due to the demands of the CPU to execute device driver, kernel I/O code; context switches due to interrupts; data copying; and network traffic” [5]. This is confirmed through

section 2.3 regarding firmware drivers. Sub-optimal I/O performance is commonly caused by hardware errors. Two of the most common causes of “poor I/O throughput are out of date firmware and insufficient queue length” [8]. Another cause can be due to a file system filter driver. This is because these drivers are designed to “intercept requests before they reach the file system, and performs additional processing such as anti-virus checking and encryption” [8]. System performance can be improved through utilizing blocking and non-blocking I/O.

2.5 Lower Cost Tablets

In order to create the most comparable tablet, the researcher has analyzed the benefits of different CPU core architecture components. This has been done in order to offer the most benefit, yet lowest cost in terms of resultant prototype tablet.

While analyzing the different components, the researcher will be able to determine the benefits and disadvantages of each option. This will, in turn, allow the researcher to determine what component would best meet the expectations of the customer, as well as be cost effective.

It is noted that “for the design of portable systems, power consumption is as important a design criterion as performance. Low power design starts at the system level and a top down approach will yield the greatest results” [10]. There were three different low cost CPU architecture choices to choose from. These were: (1) Intel architecture; (2) ARM architecture; and (3) MIPS architecture. Intel architecture was found to be incompatible because it was the most complicated and had a higher power consumption CPU. Although currently used in most low cost tablets, ARM technology has become more and more complicated, thereby making it incompatible. Finally, it was found that MIPS architecture is the simplest of all three types and has the lowest cost.

Importantly, “MIPS Technologies has produced a new mainstream processor core every three years. The company does not plan to release a major redesign of the R10000 for quite some time. In the interim, the R10000 will go through process shrinks and design tweaks, much as the R4000 did, to further increase

performance, but the long-term competitiveness of the MIPS high end rests entirely with the R10000. The R10000 sets new standards for instruction level parallelism, allowing the function units to take advantage of the parallelism inherent in the code rather than enforcing some arbitrary program ordering. Techniques such as register renaming and out of order execution will be used widely in future processor designs. The new CPU promises to restore vitality to the high end of the MIPS processor line. To fulfill this promise, however, working parts must be fabricated" [11]. These improvements have had some effects. For instance, "these incremental improvements will give high-end MIPS customers significantly better performance, particularly on these larger applications. The R12000 is likely to lag the SPEC ratings of leading processors from HP and Digital in 1H98, however. SGI expects faster versions of the R12000 to appear as early as 1999, but the next big jump in performance, particularly for bandwidth-limited applications, won't come until the processor known as the H2 appears" [12].

Importantly, "the R10000 is an aggressive out-of-order design that has a 32-entry "active list" that keeps track of all instructions that have been dispatched but not retired; three 16-entry instruction queues, one each for integer, floating-point, and memory instructions; and integer and floating-point register files, each with 64 re-namable registers. The queues are misnamed, for instructions can be issued from anywhere in the queue rather than in a first-in, first-out manner" [12].

It was found that MIPS32 released two sets of instructions for its core implementation and the following application specific extensions (ASEs): (1) The MIPS® DSP Application-Specific Extension (ASE) is optimized for signal processing applications; (2) The MIPS16e™ Application-Specific Extension (ASE) is optimized for code compression; and (3) The 74K core achieves its highest performance through the implementation of advanced superscalar and out-of-order dispatch techniques in a deeply pipelined implementation of the MIPS32 architecture. The superscalar dispatch allows the core to dispatch two instructions per cycle to two pipelines: (1) A 15-stage AGEN pipeline that executes all load/store and control transfer instructions (2) 14-stage ALU pipeline that executes all the rest of the instructions (arithmetic, logic, and general computations).

It was found that the out of order approach selected from a pool of instructions for dispatching allows each pipeline to operate independently. Thus, in order to ensure the availability of two dispatched instructions, twice the number of instructions is fetched every cycle from an instruction cache. Importantly, the 74K core also implements sophisticated branch prediction techniques that minimize the cost of an unpredicted branch in a deeply core pipeline. The 74K core information and data caches are configured as 0, 16, 32, or 64 KB in size. Each cache is organized as a 4-way set associative data structure. Importantly, the 74K core supports pre-fetching of sequential cache lines on information cache miss. Thus, the extent of pre-fetching can be configured via software to pre-fetch between 0 and 2 additional lines. The data cache features non-blocking load misses and has the capacity to handle up to 9 outstanding load misses in up to 4 unique cache lines. On a cache miss, the processor cannot continue executing instructions while the load data is being fetched, until a dependent instruction is reached. Therefore, both caches are virtually indexed and physically tagged.

ATM7019 is a high performance, lower power, and highly integrated SOC (system on chip), which targets personal media players (PMPs), internet/home media players, and is, overall, a good compatibility for multimedia. This CPU is being incorporated with power video and audio processes, which enables it to provide full HD video and high audio decoding / encoding abilities. Furthermore, the CPU offers graphic and video scaling, and it is important to note that alpha blending processes are enabled with a powerful display engine. It is noted that high performance DDR2/DDR3 controllers guarantee that there is high data throughput available for video/audio playing. In addition, ATM7019 has 16 million color LCD and touch panel interfaces to support portable media play. Thus, audio/video files can also be played through the HDMI transmitter, CVBS encoder, and YPbPR/YCbCR channels at home. Significantly, the integrated USB 2.0 with OTG functions enables the platform to act as a host or a slave mass storage device.

The 74KTM core from MIPS Technologies is reputed to be a high performance, low power 32–bit MIPS RISC processor core family and is intended for use in custom system on silicon applications. Thus, the core is designed for semiconductor manufacturing companies, ASIC developers, and system OEMS that are

required to rapidly integrate their own custom logic and peripherals with a high-performance RISC processor.

The study was very narrow in scope in order to ascertain the new innovation of a low cost tablet. Thus, it was concerned with the aspects related to the construction of a navigational scheme utilizing innovative ideas. The secondary goal of the research was to determine the best way to make cheap and affordable tablets for the Thai community. It was believed that although the tablets should be cheap and affordable, they must also be high quality products in order to provide the best product for the cost. In order to find the necessary information for the creation of a low cost, high performance tablet, it is necessary to have entirely transparent and intuitive search processes. The study shows that while navigation is important, many other factors need to be reviewed in order to provide optimal output as the best low cost, high performance tablet.

Most significantly, the research showed that there is a correlation between engagement, motivation, and attitude within the creation and initiation of this tablet. It is expected that future studies on the CPU core and architecture will be beneficial in understanding the connection between the software and CPU versus hardware and I/O infrastructure to obtain the best performance out of a lower cost tablet.

2.6 Related Studies

The final section of the literature review discusses various studies and articles related to low cost tablets. The first four studies discuss the importance of the market on tablet sales and different mechanisms. The remaining studies discuss I/O optimization.

2.6.1 India’s Aakash Project

India created the Aakash in 2011, which has been “advertised as the world’s cheapest tablet computer and is tagged with an introductory price of US $35” [13]. Since this time, India has expanded to create the second-generation Aakash (known as Aakash 2). Although the Aakash had Wi-Fi capabilities, the Aakash 2 had

mobile broadband capabilities “and came equipped with a slot for a SIM card” [13]. Furthermore, the Aakash 2 has video streaming capabilities. In contrast, it does not have external speakers. In order to meet these needs, the Aakash 2 has “a 3.5-mm jack for earphones and supporting voice input with a built-in microphone” [13]. The Aakash 2 is known for its extensive features, yet light weight of 350 grams. In fact, “the 7-inch Aakash 2 appears to fit the market standard and yet costs a fraction of its competitors” [13]. Thus, the Aakash is considered to be ‘jugaad,’ which means resourceful. This concept is positive and associated with innovative activities [13]. Thus, the Aakash and Aakash 2 “is said to be motivated by the need for a computing device with a low cost but rich in features” [13]. However, the tablet is “unlike the typical technology product that emerges from a standard market development and production cycle [in that it] was conceived out of a government tender” [13]. Common approaches to create a low-end and low-cost computer have been to “strip a machine to the most basic configurations and getting rid of operating system costs by replacing Windows with a free distribution of Linux and by including in the published price of a low-cost computer, the assumption of a state subsidy and very large sales volumes” [13]. This is not a new approach and has been used in Brazil with the Computador Popular project during the early 2000s. Thus, these types of projects focus on the “creation of a technology around a price point” [13]. The heart of these projects is to “bridge the digital divide” [13].

However, it was found that the Aakash 2 was manufactured in China, a major rival of India. Since the Aakash “had been conceived as a ‘nationalistic dare,’” this was seen as a betrayal by the government [13]. Making matters worse, the Aakash featured a ‘Made in India’ stamp. Since the project is governmental, it is believed that “it is the money of Indian taxpayers and is to be used for Indian students. It is for the government to decide whether they want to spend it for Indian jobs or they want to spend it on Chinese jobs” [13]. Despite this, it was believed that China was more adept in being efficient in creating electronic devices [13]. Finally, the article notes that “there is much to be said about technology getting cheaper. Everywhere in the world, computing technology becomes a domestic artifact only after the price hits a certain affordable point. But these devices entered various societies at points when much of other human infrastructure was already in place” [13].

2.6.2 Global Flagships and Suppliers

In 2004, China established itself as “the world’s largest producer of computer hardware” [14]. However, it is suggested that Chinese companies have low levels of contributions within the sector. However, one of the most impactful “segments of China’s burgeoning electronics industry is notebook computers” [14]. However, “Taiwan has been a major player in the production of notebook computers for many years” [14]. After 2001, most of the production activities moved to mainland China. As a result of these activities, notebook computers were “the leading electronics exports, which represented a massive boost to China’s position in the supply of IT hardware” [14]. As a result, it is suggested that “few or no linkages exist between any of the Taiwanese notebook firms and indigenous electronics firms in mainland China” [14]. In contrast, it is found that “in China there is a notebook segment where production is dominated by Taiwanese original equipment manufacturers and especially original design manufacturers with no linkages to indigenous Chinese firms” [14].

It is found that “production commonly takes place within a network of relationships organized around a flagship company with varying ‘tiers’ of subcontractors as well as relationships with other service providers and strategic alliances” [14]. Flagships are commonly brand leaders and/or contract manufacturers. These two companies work together. Lead firms derive power from “control over critical resources and capabilities that facilitate innovation and from their capacity to coordinate transactions and knowledge exchange between the two network nodes. Flagships retain in-house activities which give them a particular strategic advantage, and outsource the ones that do not” [14]. Local suppliers can be classified as higher- and/or lower-tier suppliers. While “higher-tier suppliers deal directly with global flagships and play an intermediary role between global flagships and lower-tier suppliers, lower-tier suppliers rarely interact directly with the global flagships; they deal primarily with local higher-tier suppliers” [14]. One of the primary reasons for investing in China is cost-savings for many foreign firms. However, if considering logistic costs, “offshore manufacturing in China does not make too much of a cost difference” [14]. Ultimately, these types of relationships between flagships and suppliers allow low-cost tablets to be built.

2.6.3 Xiaomi Tablets

Although the company started with smartphones, it has expanded into tablets “with a new low-price slate set to rival Apple’s iPad mini” [15]. Since the company began with Android smartphones, it has experienced “big sales and a growing fan base … [Therefore,] Xiaomi is taking the same strategy with tablets and unveiled an Android product starting at 1499 yuan (US$243) … The Xiaomi tablet has a 7.9-inch screen with a resolution of 2048 by 1536 pixels. It uses Nvidia’s new Tegra K1 processor, a 2.2 GHz quad-core chip. There are two versions, with either 16GB or 64GB of internal storage, and both have a microSD card slot for up to 128GB of additional storage” [15]. It is remarkably cheaper than the iPad Mini, which starts at 2888 yuan. However, there are concerns related to the late start that the company has in the tablet market. This is because “Apple has reigned as the nation’s market leader, and in the first quarter had a 43 percent share. Samsung trailed in second with a 10 percent share, while homegrown PC maker Lenovo had 5 percent” [15].

2.6.4 Apple and Samsung in china

Apple is known as being the “world’s largest company with nearly $600 billion in market value” [16]. However, originally, when Apple entered the Chinese market, it was the “underdog” [16]. As of 2012, there were “five Apple stores in mainland China” [16]. However, there is a war taking place between Apple and Samsung in China. In this scenario, “Korea’s Samsung is firmly entrenched with 18% of the smartphone market vs. Apple’s 12%. Furthermore, Samsung has indicated it will amp up an already large China-centric marketing and advertising budget, which could put pressure on Apple to do the same” [16]. Other issues involve Apple’s narrow product line in comparison to Samsung’s wide product line. Thus, “with this diverse product line, Samsung has relationships with tens of thousands of retailers and distributors, including venders in so-called Tier 5 cities, those with around 500,000 residents” [16]. Significantly, “for all Apple’s gains in China, the country remains a relatively small contributor to the device maker’s revenue mix. Of its $108 billion in 2011 revenue, only 12% came from China … Samsung currently gets about 30% of its overall revenue from China. This issue is front and center for Apple and for Wall

Street. Toni Sacconaghi, Sanford Bernstein's Apple analyst, estimates that 70% or more of the smartphones currently purchased in China sell for $300 or less" [16].

Research shows that "China's big three electronics makers are also making aggressive bids for the tablet market, where Apple is the current market-share leader. The iPad 2 sold only moderately well in China, and, again, like the iPhone, it is a one size fits all solution. Lenovo, ZTE, and Huawei offer a range of tablet-like devices, from inexpensive models that look a bit like oversize phones to Lenovo's new Yoga line, which combines a touchscreen with a full keyboard. All of which make rumors of an iPad mini especially interesting in the Chinese market. Several reports suggest the new device will boast a seven-inch screen and retail in the U.S. for about $300. This lower-cost tablet could be the kind of entry-level device a big swath of China has been waiting for: a chance to take a bite of the Apple brand, without investing half-a-year's salary. It might even suggest a willingness to develop entry-level products for the iPhone and computing lines. Meanwhile Apple is moving in smaller, smart ways to further immerse itself in the China market. The new iOS 6 operating system integrates popular sites in China like Sina Weibo, the microblog that's more or less the equivalent of Twitter. And the Mac OS 10.8 upgrade includes a package of popular Mandarin sites, including Youku, a video destination. And while it lacks the massive reach of its Chinese competitors and Samsung, Apple has dramatically expanded the number of stores it permits to sell its devices; there are 11,000 places in China to buy the iPhone, up 138% from last year. The aggressive expansion plainly continues" [16].

2.6.5 I/O Subsystem Optimization

According to the study, the optimizations are accomplished "with the support of new hardware interfaces" [17]. The optimizations are designed to "minimize per-request latency by streamlining the I/O path and amortize per-request latency by maximizing parallelism" [17]. Through the study, it was found "that eliminating context switches in the I/O path decreases the software overhead of an I/O request from 20 microseconds to 5 microseconds and a new request merge scheme called Temporal Merge enables the OS to achieve 87% to 100% of peak device performance, regardless of request access patterns or types" [17]. There are six options for block I/O subsystem optimizations: "(1) polling I/O completion; (2) eliminating context switches from the

I/O path; (3) merging discontinuous requests; (4) reconfiguring an I/O scheduler for an SSD; (5) resolving the read-ahead dilemma; and (6) avoiding a lock contention in a request queue" [17]. The study found that "the block I/O subsystem without any context switches in the I/O path loses a chance to accumulate requests in an I/O scheduler" [17]. The study also found that "it is essential to have both a new hardware interface and an optimized block I/O subsystem to fully exploit such latency and throughput of the device under various workloads. The most important lesson learned from the evaluation is that a block I/O subsystem should part from disk-based optimizations and reexamine any software overhead at microsecond granularity" [17].

2.6.6 Samsung Microprocessor Chips

In 2013, TSMC was recruited by Apple "to produce its future a series processor chips while undergoing legal battles with Samsung, their current provider" [18]. Thus, TSMC produced most of the A8 processors. However, Apple has reportedly "turned back to Samsung to manufacture its A9 chip" [18]. This chip is produced using "14nm FinFET technology" [18]. While Samsung creates chips for Apple, it is "putting significant pressure on Qualcomm with the pervasive rumors that Samsung will use its own microprocessors in the next version of the Galaxy S smartphone" [18]. These microchips are "extremely thin ePoP (embedded package on package) memory, a single memory package consisting of 3GB LPDDR3 DRAM, 32GB eMMC and a controller for use in high-end smartphones. The 3GB LPDDR3 mobile DRM inside the ePOP operates at an I/O data transfer rate of 1866MB/s, with a 64-bit I/O bandwidth. The chip also features special heat-resistant properties" [18]. It is believed by Samsung that the new chip will "decrease the total area used by approximately 40%" [18].

2.7 Theoretical, Conceptual, and Operational Frameworks

There are different theories that affect computer science. The goal of this section is to discuss these theories in order to develop the theoretical, conceptual, and operational frameworks.

2.7.1 Theoretical Framework

The theory of computation “can be divided into the following three areas: complexity theory, computability theory, and automata theory” [19]. The goal of complexity theory is to “classify problems according to their degree of ‘difficulty’” [19]. The goal of computability theory is to “classify problems as being solvable or unsolvable” [19]. The goal of automata theory is to determine whether or not “these models have the same power, or if one model can solve more problems than the other” [19].

For this study, the computability theory is being used. This is because problems are classified in one of two ways. Since the purpose of the study is to determine how I/O, IRQ, and firmware drivers can best influence performance, the results of these components can be defined as “solvable” or “unsolvable.”

2.7.2 Conceptual Framework

The goal of this figure is to show the factors affecting tablet creation. The conceptual framework is shown below:

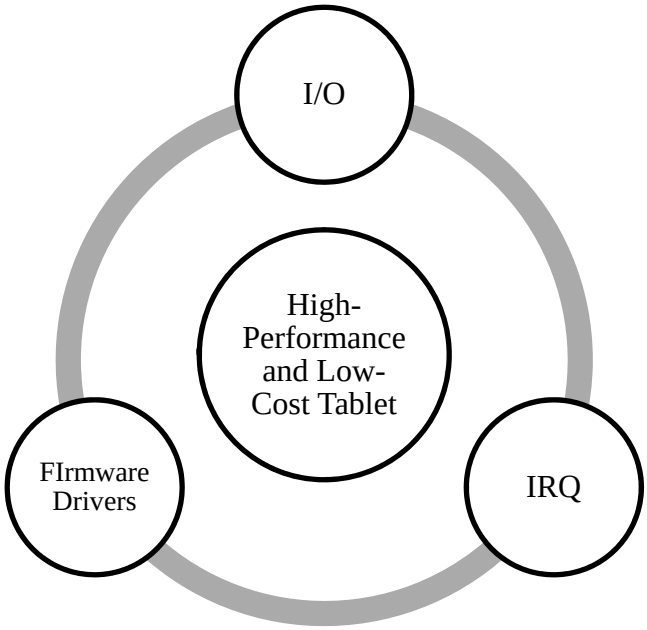

Figure 2.5 Conceptual Framework.

2.7.3 Operational Framework

However, it is hypothesized that the theory can be expanded to define the components as “efficient” or “inefficient.” This can lead to hypotheses and null hypotheses related to the study, which can then be analyzed utilizing the Chi Square Test. The goal of this figure is to show how efficiency can be measured with each variable. The operational framework is shown below:

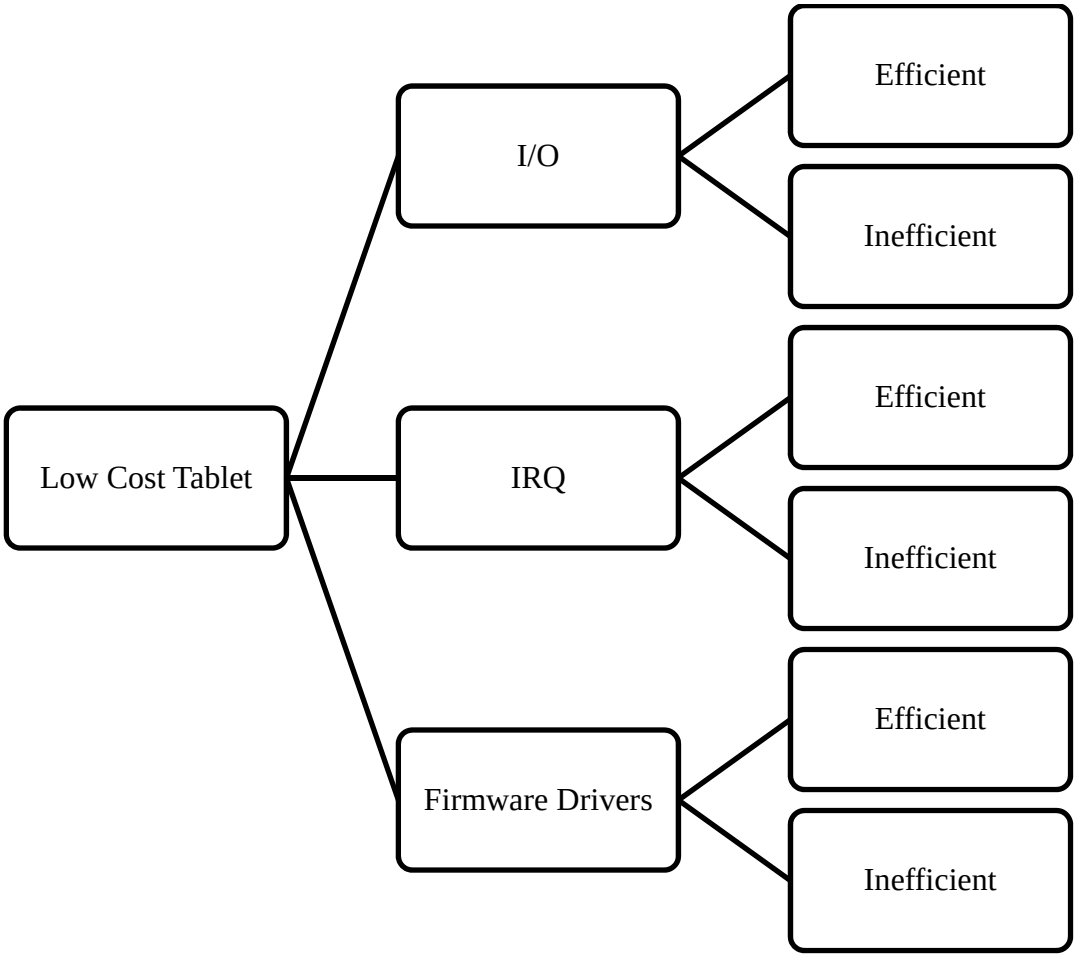

Figure 2.6 Operational Framework.

CHAPTER III
RESEARCH METHODOLOGY

The goal of this chapter is to describe the research design and analysis method. The chapter is divided into two separate sections: variables and measurement; and research design and analysis.

3.1 Variables and Measurement

The conceptual framework shows how I/O, IRQ, and firmware drivers interact to create a low cost, high performance tablet. Therefore, the measurements are based on I/O, IRQ, and firmware drivers. Each of these three variables is measured as being efficient or inefficient. For the purpose of this study, the I/O speeds are found in Table 1, replicated below:

Table 3.1 I/O Speeds.

Device	**Data Rate**
Keyboard	10 bytes / sec
Mouse	100 bytes / sec
56K Modem	7 KB / sec
802.11g Wireless	6.75 MB / sec
52x CD-ROM	7.8 MB / sec
Fast Ethernet	12.5 MB / sec
Compact Flash Card	40 MB / sec
FireWire (IEEE 1394)	50 MB / sec
USB 2.0	60 MB / sec
SONET OC-12 Network	78 MB / sec
SCSI Ultra 2 Disk	80 MB / sec
Gigabit Ethernet	125 MB / sec
SATA Disk Drive	300 MB / sec
Ultrium Tape	320 MB / sec
PCI Bus	528 MB / sec

Each of these rates will be considered for the other two variables: IRQ and firmware drivers. For the purposes of this study, IRQ is determined through non-maskable interrupts. These are shown in the following table:

Table 3.2 Maskable Interrupts.

Vector Number	Description
1	Debug exception
2	Null intercept
3	Breakpoint
4	INTO-detected outflow
5	Bound range exception
6	Invalid opcode
7	Device not available
8	Double fault
9	Coprocessor segment overrun (reserved)
10	Invalid task state segment
11	Segment not present
12	Stack fault
13	General protection
14	Page fault
15	Reserved, do not use
16	Floating-point error
17	Alignment check
18	Machine check
19 – 31	Reserved, do not use

Since most computers need 500 MB free to run, this is being used for driver speed. Based on the above information, the following equation will be used to determine initial efficiency:

$$Efficiency = \sqrt{\frac{\bar{A} \times \bar{B} \times \bar{C}}{N}}$$

Ten different architectures are measured in terms of efficiency. The results from the efficiency measurements define *N*, while *A* is defined by averaged I/O data rate; *B* is defined by averaged IRQ vector; and *C* is defined by averaged driver speed. The architectures being measured include:

1) OMAP3640
2) QSD8250
3) Tegra2
4) i.MX515
5) TCC8803
6) WM8650
7) SSPV210
8) RK2918
9) A10 / A13
10) ATM709

Each of these architectures is measured considering performance and cost. The highest rate any architecture could obtain was 10, while the lowest was 2. Individually, each factor was scored on a scale between 1 and 5, with 5 being the highest score possible. These results are shown below:

Table 3.3 CPU Research and Scores.

CPU	Company / Country	Frequency	Arch	Perform	Cost	Total
RK2918	Rock chip / China	1 G – 1.2 G	ARM Cortex A8	4.5	4.5	9
A10 / A13	All winners / China	1 G	ARM Cortex A8	5	4.5	9.5
ATM7019	Actions-Semi / China	1 G	MIPS	5	5	10
S5PV210	Samsung / Korea	1 G	ARM Cortex A8	5	4	9
OMAP3640	TI / USA	1G	ARM Cortex A8	4	3	7
WM8650	VIA / Taiwan	600 MHz + 300 MHz DSP	ARM11 + DSP	4	4.5	8.5
QSD8250	Qualcomm / USA	1 G	ARM / Snapdragon	4.5	3	7.5
Tegra2	NVidia / USA	1 G	ARM Cortex A9	5	3	8
TCC8803	Tele chips / Korea	1 G	ARM Cortex A8	4.5	4	8.5
i.MX515	Freescale / USA	800 MHz	ARM Cortex A8	4	4	8

The goal of this table was to determine what CPU architecture would be most efficient. Since two categories performance and cost were measured, the highest possible score that any design could obtain was 10, whereas the lowest possible score that any design could obtain was 2. The CPU architecture ranked (from least to highest). This information will be utilized in a statistical analysis to compare the effects of each of the variables.

3.2 Research Design and Analysis

As noted previously, the variables are I/O, IRQ, and firmware drivers. Upon further consideration, the researcher elected to measure the efficiency of 10

different CPU architectures, shown in Table 5 in the preceding section. This is being used as a dummy variable, N, in the analysis. This is being done to stabilize the results. Variable A is defined by averaged I/O data rate; variable B is defined by averaged IRQ vector; and variable C is defined by averaged driver speed. This is used to calculate an efficiency rate of:

$$Efficiency = \sqrt{\frac{\bar{A} \times \bar{B} \times \bar{C}}{N}}$$

This calculated efficiency rate will be compared to the already established efficiency rate from Table 5. This will be used to conduct a Chi Square Test analysis to determine the rate of efficiency for the different variables.

The Chi Square Test is important because it shows whether or not "a relationship observed in a sample that would have been unlikely to occur if really there is no relationship in the larger population" [20]. The most common significance is $p < 0.05$, so it will be used in this scenario as well.

The efficiencies can be compared in a scatter graph and used to conduct a linear regression analysis. With this information, further relationships can be determined between the variables. This will include measures of central tendency in order to determine average efficiencies for each of the variables.

CHAPTER IV
RESULTS AND DISCUSSION

The chapter opens with the numerical analysis of each variable, including the dummy variable. The fifth section shows all variables combined. The sixth section shows the efficiency test developed in the preceding chapter. The seventh section shows the comparison of both efficiencies through linear regression (including measures of central tendency). The final section shows the comparison of the three variables to the dummy variable in a Chi Square Test.

4.1 I/O

The data rate for the I/O devices is not the same. For instance, some are in bytes, while some are in kilobytes. For simplicity, all results are being converted to megabytes. The numerical values for the I/O variable (A) are:

Table 4.1 I/O Variable (A).

Device	Data Rate	Conversion (to MB)
Keyboard	10 bytes / sec	0.000009537
Mouse	100 bytes / sec	0.000095367
56K Modem	7 KB / sec	0.006835937
802.11g Wireless	6.75 MB / sec	6.75
52x CD-ROM	7.8 MB / sec	7.8
Fast Ethernet	12.5 MB / sec	12.5
Compact Flash Card	40 MB / sec	40
FireWire (IEEE 1394)	50 MB / sec	50
USB 2.0	60 MB / sec	60
SONET OC-12 Network	78 MB / sec	78
SCSI Ultra 2 Disk	80 MB / sec	80
Gigabit Ethernet	125 MB / sec	125
SATA Disk Drive	300 MB / sec	300
Ultrium Tape	320 MB / sec	320
PCI Bus	528 MB / sec	528

4.2 Interrupt

The full table has 256 entries and is 1024 bytes. Therefore, for this study, the vector number will be multiplied by bytes and converted to megabytes for consistency. The numerical values for IRQ are found below:

Table 4.2 IRQ Variable (B).

Vector	Bytes	IRQ (*B*)
1	1024	0.000977
2	2048	0.001953
3	3072	0.00293
4	4096	0.003906
5	5120	0.004883
6	6144	0.005859
7	7168	0.006836
8	8192	0.007813
9	9216	0.008789
10	10240	0.009766
11	11264	0.010742
12	12288	0.011719
13	13312	0.012695
14	14336	0.013672
15	15360	0.014648
16	16384	0.015625
17	17408	0.016602
18	18432	0.017578
19	19456	0.018555
20	20480	0.019531
21	21504	0.020508
22	22528	0.021484
23	23552	0.022461
24	24576	0.023438
25	25600	0.024414
26	26624	0.025391
27	27648	0.026367
28	28672	0.027344
29	29696	0.02832
30	30720	0.029297
31	31744	0.030273

4.3 Firmware Driver

In order for most CPUs to run efficiently, they need 500 MB of RAM free. This is being used as the firmware driver variable C.

4.4 Dummy Variable

The numerical values for the dummy variable are shown below:

Table 4.3 Dummy Variable (N).

Variable	CPU	Total
N1	RK2918	9
N2	A10 / A13	9.5
N3	ATM7019	10
N4	S5PV210	9
N5	OMAP3640	7
N6	WM8650	8.5
N7	QSD8250	7.5
N8	Tegra2	8
N9	TCC8803	8.5
N10	i.MX515	8

4.5 Variables Combined

For ease of study, the previous tables are combined to have all values easily found in one centralized location:

Table 4.4 Variables A, B, C, and N.

Variable A		Variable B		Variable C	Variable N	
A1	0.000009537	B1	0.000977	500	N1	9
A2	0.000095367	B2	0.001953	500	N2	9.5
A3	0.006835937	B3	0.00293	500	N3	10
A4	0.390625	B4	0.003906	500	N4	9
A5	3.5	B5	0.004883	500	N5	7
A6	6.75	B6	0.005859	500	N6	8.5
A7	7.8	B7	0.006836	500	N7	7.5
A8	12.5	B8	0.007813	500	N8	8
A9	40	B9	0.008789	500	N9	8.5
A10	50	B10	0.009766	500	N10	8
A11	60	B11	0.010742	500		
A12	78	B12	0.011719	500		
A13	80	B13	0.012695	500		
A14	125	B14	0.013672	500		
A15	300	B15	0.014648	500		
A16	320	B16	0.015625	500		
A17	528	B17	0.016602	500		
		B18	0.017578	500		
		B19	0.018555	500		
		B20	0.019531	500		
		B21	0.020508	500		
		B22	0.021484	500		
		B23	0.022461	500		
		B24	0.023438	500		
		B25	0.024414	500		
		B26	0.025391	500		
		B27	0.026367	500		
		B28	0.027344	500		
		B29	0.02832	500		
		B30	0.029297	500		
		B31	0.030273	500		
Average	94.82044505		0.015625	500		

4.6 Efficiency Test

The efficiency test is being conducted using the following formula:

$$\sqrt{\frac{\bar{A} \times \bar{B} \times \bar{C}}{N}}$$

In the formula, variable A refers to the average of variables A1 to A17. Variable B refers to the average of variables B1 to B31. Variable C refers to 500 MB. Variable N refers to variables N1 to N10. These values are all found in section 4.5. The formulas will be calculated based on the dummy variables. This results in the following calculations:

Table 4.5 Results of Efficiency Calculations.

Architecture	Established Efficiency	Tested Efficiency
N1	9	9.1
N2	9.5	8.8
N3	10	8.6
N4	9	9.1
N5	7	10.3
N6	8.5	9.3
N7	7.5	9.9
N8	8	9.6
N9	8.5	9.3
N10	8	9.6
Average	8.5	9.4

This shows that I/O, IRQ, and firmware drivers have different impacts on performance.

4.7 Comparison through Chi Square Test

The study wants to determine the influence of I/O, IRQ, and firmware drivers on efficiency. This can be done using hypotheses and null hypotheses, as well as the data already obtained. This study has used a dummy variable for a consistent variable and calculated a new efficiency variable for each of the architectures (found in Table 11). Therefore, these are the two factors used to conduct the Chi Square Test.

The first section involves establishing the hypothesis and null hypothesis. In this case, the hypothesis is: I/O, IRQ, and firmware drivers increase efficiency in CPUs. The null hypothesis is: I/O, IRQ, and firmware drivers do not increase efficiency in CPUs.

The second section involves determining the Chi Square. This is done by completing the following table:

Table 4.6 Chi Square Table.

Observed		**Total**	**Expected**		**Chi Square**	
9.0	9.1	18.1	8.6	9.5	0.017	0.016
9.5	8.8	18.3	8.7	9.6	0.072	0.065
10.0	8.6	18.6	8.9	9.7	0.149	0.135
9.0	9.1	18.1	8.6	9.5	0.017	0.016
7.0	10.3	17.3	8.2	9.1	0.185	0.168
8.5	9.3	17.8	8.5	9.3	0.000	0.000
7.5	9.9	17.4	8.3	9.1	0.074	0.067
8.0	9.6	17.6	8.4	9.2	0.017	0.015
8.5	9.3	17.8	8.5	9.3	0.000	0.000
8.0	9.6	17.6	8.4	9.2	0.017	0.015
85.0	93.6	178.6			1.045	

This table provides the Chi Square in order to prove or disprove the hypothesis or null hypothesis. The expected numbers are calculated by multiplying the corresponding row and column, then dividing this value by the total. The Chi Square is calculated by finding the square of the difference between the observed and expected values, then dividing this number by the expected value. This is done for each value,

then added together to get a final Chi Square number. The third section involves calculating the degrees of freedom as follows:

$$df = (Number\ of\ Rows - 1) \times (Numbers\ of\ Columns - 1)$$
$$= (10 - 1) \times (2 - 1) = 9$$

Thus, with a significance of $p < 0.05$, the critical value is 3.940. In this case, if the Chi Square is lower than the critical value, the null hypothesis is rejected. Since the Chi Square is 1.045 and is lower than the critical value of 3.940, the null hypothesis is rejected. Therefore, it is confirmed that I/O, IRQ, and firmware drivers increase efficiency.

4.8 Discussion

This research study is important especially considering the popularity of the iPad. In fact, during the first quarter of 2014, there were 26.04 million iPads sold globally [1]. Furthermore, most companies that develop tablets are focused on making a profit. As a result, these companies are focusing on integrating new technological designs in order to create a high performing, yet low cost tablet. This focus has resulted in numerous tablets available for purchase, such as the: iPad Air 2, iPad Air, iPad mini 2, Samsung Galaxy Tab S 10.5, and Samsung Galaxy Tab S 8.4 [2].

Within the top ten list, five of the tablets belonged to either Samsung or Apple. Part of this was due to the fact that both companies design their own CPUs. For instance, in early 2015, Samsung “began developing a custom CPU core project for mobile application processors and is considered to be an important step by the company because it can establish Samsung Semiconductor as a maker and a supplier of grade-a silicon for mobile device vendors” [3]. For Samsung to meet its organizational goals, this is an unsurprising move. However, at the same time, it is incredible because Apple hired one of the top engineers and designers from Samsung. This move was important for Apple because it solidified “Apple’s commitment to designing its own chips” [4]. The company had been customized chips built on ARM architecture. However, the A6 chip (used in the iPhone 5) was the first chip to be built “from the ground up using just

ARM's reference designs, not its ready-made cores" [4]. This is considered to be a significant accomplishment within the chip design world.

Since Samsung and Apple design their own chips, they have an advantage over other companies in that they are able to control where the I/Os are placed on the circuit board, allowing them to create optimal efficiency and output. Thus, interrupt requests specification and selections from the drivers in the firmware is important for the CPU to specify the I/O correctly in order to obtain optimized output from the I/Os. However, "programmed I/O (PIO) refers to data transfers initiated by a CPU under driver software control to access registers or memory on a device.

The CPU issues a command then waits for I/O operations to be complete. As the CPU is faster than the I/O module, the problem with programmed I/O is that the CPU has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The CPU, while waiting, must repeatedly check the status of the I/O module, and this process is known as Polling. As a result, the level of the performance of the entire system is severely degraded. The CPU issues commands to the I/O module then proceeds with its normal work until interrupted by I/O device on completion of its work. For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports, memory mapping. For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer. Although Interrupt relieves the CPU of having to wait for the devices, but it is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory." [6].

Thus, the objective of this study was to explore how IRQs and firmware drivers make I/Os work better to create cheaper tablets. Since the study is considering custom-made chips, the scope revolves primarily around Samsung and Apple, but also some local vendors. This includes chip design, IRQ, and firmware drivers. The scope also includes maximum output by I/Os. Based on this study, it is expected that through knowing the exact location of I/O placement on the PC board for maximum output,

Apple and Samsung will be able to offer cheaper tablets that work better than other tablets that do not have customized I/Os.

Although the goals of I/O vary, all computers (and tablets) must have I/O in order to be useful, resulting in varying I/O speeds. For instance, a keyboard's data rate is 10 bytes per second, whereas a PCI bus's data rate is 528 MB per second [5], [11]. However, all I/Os have operational parameters, which include: byte/block, sequential/random, and polling/interrupts. Since the I/O devices have different outputs, the goal of the I/O subsystem is to "provide uniform interface for a wide range of devices" [5].

Ultimately, the "CPU interacts with the device controller" [5]. This controller may contain different read/write registers and memory. Therefore, information can be accessed in two different ways by the processor. The first way is through I/O instructions. The second way is through memory mapped I/O. To begin with, I/O instructions contains "explicit in/out instructions" [5]. Second, memory mapped I/O loads and stores information. There are differences in I/O instructions and memory mapped I/O. For instance, I/O instructions utilize assembly language and "prevents user-mode I/O" [5]. Memory mapped I/O does not require special instructions (explicit instructions) and can be used in C, allowing for "user-based I/O" [5]. However, caching addresses must be prevented in memory mapped I/O [5]. Furthermore, "Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module controls exchange of data between main memory and the I/O device. Because of DMA device can transfer data directly to and from memory, rather than using the CPU as an intermediary, and can thus relieve congestion on the bus. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred. Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles. DMA increases system concurrency by allowing the CPU to perform tasks while the DMA system transfers data via the system and memory busses. Hardware design is complicated because the

DMA controller must be integrated into the system, and the system must allow the DMA controller to be a bus master. Cycle stealing may also be necessary to allow the CPU and DMA controller to share use of the memory bus" [6].

The application I/O interface varies. That is, devices can vary in terms of dimensions to include "character-stream or block, sequential or random-access, sharable or dedicated, speed of operation, and read-write, read only, or write only" [5]. As a result, there is a device-specific code within the kernel. This code is designed to "interact directly with the device hardware" [5]. Thus, the kernel is important because it "keeps state for I/O components, including open file tables, network connections, and character device state" [5].

Device drivers are composed of the top and bottom half. The top half is designed to start I/O to the device and "is accessed in [the] call path from system cells" [5]. However, the top half may also "put the thread to sleep until it is finished" [5]. The bottom half "runs as an interrupt routine" [5]. Thus, the top half leads the bottom half in the undertaken functions. The different devices include block and character devices, network devices, clock devices and timers, and clock hardware.

OS mechanisms involve functions such as data transfer, notification and buffering. Thus, programming I/O involves transferring data to/from the controller by transferring "each byte via processor in/out or load/store" [5]. As a result, programmed I/O is easy to use/program and is simple. However, programmed I/O is controversial because it "consumes processor cycles proportional to data size," which can hinder performance [5]. If the system uses direct memory access, data is transferred to/from the controller by "giving the controller access to the memory bus and asking it to transfer data to/from memory directly," making it beneficial for large data movement [5]. However, direct memory access is controversial because it 'requires a DMA controller and bypasses the CPU to transfer data directly between the I/O device and memory" [5].

Research also shows that "since input and output devices are not under the full control of the CPU (I/O events are asynchronous), the CPU must somehow be told when an input device has new input ready to send, and an output device is ready to receive more output. The FGI flip-flop is set to 1 after a new character is shifted into INPR. This is done by the I/O interface, not by the control unit. This is an example of

an asynchronous input event (not synchronized with or controlled by the CPU). The FGI flip-flop must be cleared after transferring the INPR to AC. This must be done as a micro operation controlled by the CU, so we must include it in the CU design. The FGO flip-flop is set to 1 by the I/O interface after the terminal has finished displaying the last character sent. It must be cleared by the CPU after transferring a character into OUTR" [7].

Polling is useful because it shows when the operation is complete or an error is encountered. Thus, polling occurs when the "OS periodically checks a device-specific status register or the I/O device puts completion information in the status register" [5]. Polling can be simple and have a low overhead. However, it can also waste cycles if there are "infrequent, expensive, or unpredictable I/O operations" [5]. Therefore, in some cases, an I/O interrupt is used. This occurs "whenever the I/O device needs service" [5]. Interrupts are most beneficial for unpredictable events, yet have a high overhead cost. As a result, some devices use polling and interrupts.

This information shows that I/O influences system performance due to the "demands of the CPU to execute device driver, kernel I/O code; context switches due to interrupts; data copying; and network traffic" [5]. However, system performance can improved through the use of "direct memory access; reduced/eliminated data copying; reduced number of context switches; and reduced interrupts by using large transfers, smart controllers, or polling" [5]. Furthermore, the system can use blocking and non-blocking I/O. Blocking I/O causes the process to wait. A non-blocking I/O does not cause the process to wait. An asynchronous I/O may require the process to notify the user [5]. The asynchronous I/O emphasizes the importance of the kernel I/O subsystem because it allows for interfaces designed for device reservation, caching, scheduling, and spooling [5].

Interrupts are especially beneficial for the CPU because they allow the CPU to complete its processes effectively. Thus, "the CPU issues commands to the I/O module then proceeds with its normal work until interrupted by [the] I/O device on completion of its work" [6]. Interrupt processes for input and output differ. During input, "the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor" [6]. During output, "the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer"

[6]. The goal of interrupt is to prevent the CPU from “having to wait for the devices” [6]. The basic operations of interrupt include: “CPU issues read command; I/O module gets data from peripheral whilst CPU does other work; I/O module interrupt CPU; CPU requests data; and I/O module transfers data” [6]. Furthermore, interrupts are used as an alternative to polling. In contrast to polling, interrupts are used so that “the running program is not responsible for checking the status of I/O devices” [7]. This is done through the use of an ISR. In this way, interrupts can occur at any time, provided they are enabled in the system. This allows the interrupt to interrupt the “flow of the program, execute the ISR, and then resume the program” [7]. This cause the program to be uninvolved and unaware of the interruption [7]. The primary benefit of interrupts is that “user program progress is only halted during actual transfer” [8]. However, interrupts are at a disadvantage because they require special hardware in order to “cause an interrupt, detect an interrupt, or save the proper states to resume after the interrupts” [8].

Firmware drivers are known to have an impact on I/O speed. Since I/O requests are asynchronous, “when a read or write request is made, the thread making it waits for the I/O to complete” [10]. This allows the CPU to be used for other threads. However, when the I/O request takes longer than anticipated, it can be caused by “I/O subsystem problems or misconfiguration, excessive I/O requests, data files not optimally placed on the disk, and fragmentation” [10]. Two of the most common causes of poor I/O speed are caused by “out of date firmware and insufficient queue length” [10]. Another cause can be due to a file system filter driver. This is because these drivers are designed to “intercept requests before they reach the file system, and performs additional processing such as anti-virus checking and encryption” [10].

Thus, it can be confirmed that there is a distinct relationship between I/O, IRQ, and firmware drivers. Despite the different goals for I/O, it is a necessary component of the CPU’s operations. Furthermore, the variance in I/O goals causes different devices to have different data rates [5], [11]. However, I/O has operational parameters which varies dependent upon the type of device. The relationship also confirms that since the I/O needs to meet a variety of goals, the subsystem needs to provide a uniform interface that can meet the needs of multiple devices [5].

It is noted that “for the design of portable systems, power consumption is as important a design criterion as performance. Low power design starts at the system level and a top down approach will yield the greatest results” [12]. There were three different low cost CPU architecture choices to choose from. These were: (1) Intel architecture; (2) ARM architecture; and (3) MIPS architecture. Importantly, “MIPS Technologies has produced a new mainstream processor core every three years. The company does not plan to release a major redesign of the R10000 for quite some time. In the interim, the R10000 will go through process shrinks and design tweaks, much as the R4000 did, to further increase performance, but the long-term competitiveness of the MIPS high end rests entirely with the R10000” [13]. These improvements have had some effects. For instance, “these incremental improvements will give high-end MIPS customers significantly better performance, particularly on these larger applications” [14]. Furthermore, “the R10000 is an aggressive out-of-order design that has a 32-entry “active list” that keeps track of all instructions that have been dispatched but not retired; three 16-entry instruction queues, one each for integer, floating-point, and memory instructions; and integer and floating-point register files, each with 64 re-namable registers. The queues are misnamed, for instructions can be issued from anywhere in the queue rather than in a first-in, first-out manner” [14].

It was found that the out of order approach selected from a pool of instructions for dispatching allows each pipeline to operate independently. Thus, in order to ensure the availability of two dispatched instructions, twice the number of instructions is fetched every cycle from an instruction cache. Importantly, the 74K core supports pre-fetching of sequential cache lines on information cache miss. Thus, the extent of pre-fetching can be configured via software to pre-fetch between 0 and 2 additional lines. The data cache features non-blocking load misses and has the capacity to handle up to 9 outstanding load misses in up to 4 unique cache lines. On a cache miss, the processor cannot continue executing instructions while the load data is being fetched, until a dependent instruction is reached.

ATM7019 is a high performance, lower power, and highly integrated SOC (system on chip), which targets personal media players (PMPs), internet / home media players, and is, overall, a good compatibility for multimedia. This CPU is being incorporated with power video and audio processes, which enables it to provide full

HD video and high audio decoding / encoding abilities. Furthermore, the CPU offers graphic and video scaling, and it is important to note that alpha blending processes are enabled with a powerful display engine. It is noted that high performance DDR2 / DDR3 controllers guarantee that there is high data throughput available for video / audio playing. In addition, ATM7019 has 16 million color LCD and touch panel interfaces to support portable media play. The 74KTM core from MIPS Technologies is reputed to be a high performance, low power 32 – bit MIPS RISC processor core family and is intended for use in custom system-on-silicon applications. Thus, the core is designed for semiconductor manufacturing companies, ASIC developers, and system OEMS that are required to rapidly integrate their own custom logic and peripherals with a high-performance RISC processor.

Most significantly, the research showed that there is a correlation between engagement, motivation, and attitude within the creation and initiation of this tablet. It is expected that future studies on the CPU core and architecture will be beneficial in understanding the connection between the software and CPU versus hardware and I/O infrastructure to obtain the best performance out of a lower cost tablet.

The Aakash project in India has a rich history. Originally, the Aakash was “advertised as the world’s cheapest tablet computer” [15]. The project was “motivated by the need for a computing device with a low cost but rich in features” [15]. Common approaches to create a low-end and low-cost computer have been to “strip a machine to the most basic configurations and getting rid of operating system costs by replacing Windows with a free distribution of Linux and by including in the published price of a low-cost computer, the assumption of a state subsidy and very large sales volumes” Thus, these types of projects focus on the “creation of a technology around a price point” [15]. The heart of these projects is to “bridge the digital divide” [15].

In 2004, China established itself as “the world’s largest producer of computer hardware” [16]. However, one of the most impactful “segments of China’s burgeoning electronics industry [is] notebook computers” [16]. After 2001, most of the production activities moved to mainland China from Taiwan. It is found that “production commonly takes place within a network of relationships organized around a flagship company with varying ‘tiers’ of subcontractors as well as relationships with other service providers and strategic alliances” [16]. Flagships are commonly brand

leaders or contract manufacturers. These two companies work together. Local suppliers can be classified as higher- and/or lower-tier suppliers. Ultimately, these types of relationships between flagships and suppliers allow low-cost tablets to be built.

Xiaomi established itself in the market with smartphones. However, it has since expanded into tablets by developing a tablet that will "rival Apple's iPad mini" [17]. However, the company's fan base came from the sale of Android smartphones. Thus, it is expected that the low-cost tablet, starting at 1499 yuan, will be popular within the market. The tablet has strengths in that it "has a 7.9-inch screen with a resolution of 2048 by 1536 pixels. It uses Nvidia's new Tegra K1 processor, a 2.2 GHz quad-core chip. There are two versions, with either 16GB or 64GB of internal storage, and both have a microSD card slot for up to 128GB of additional storage" and is much cheaper than the iPad Mini [17].

Apple is known as being the "world's largest company with nearly $600 billion in market value" [18]. However, there is a war taking place between Apple and Samsung in China. There is also a narrow Apple product line in comparison to Samsung's wide product line. This strong presence by Samsung has created "relationships with tens of thousands of retailers and distributors," [18]. Research shows that "China's big three electronics makers are also making aggressive bids for the tablet market, where Apple is the current market-share leader. Lenovo, ZTE, and Huawei offer a range of tablet-like devices" [18].

According to the study, the optimizations are accomplished "with the support of new hardware interfaces [and are designed to] minimize per-request latency by streamlining the I/O path and amortize per-request latency by maximizing parallelism" [19]. However, "eliminating context switches in the I/O path decreases the software overhead of an I/O request from 20 microseconds to 5 microseconds and a new request merge scheme called Temporal Merge enables the OS to achieve 87% to 100% of peak device performance, regardless of request access patterns or types" [19]. There are six options for block I/O subsystem optimizations: "(1) polling I/O completion; (2) eliminating context switches from the I/O path; (3) merging discontinuous requests; (4) reconfiguring an I/O scheduler for an SSD; (5) resolving the read-ahead dilemma; and (6) avoiding a lock contention in a request queue" [19].

Apple has reportedly “turned back to Samsung to manufacture its A9 chip” [20]. This chip is produced using “14nm FinFET technology” [20]. While Samsung creates chips for Apple, it is “putting significant pressure on Qualcomm with the pervasive rumors that Samsung will use its own microprocessors in the next version of the Galaxy S smartphone” [20]. These microchips are “extremely thin ePoP (embedded package on package) memory, a single memory package consisting of 3GB LPDDR3 DRAM, 32GB eMMC and a controller for use in high-end smartphones. The 3GB LPDDR3 mobile DRM inside the ePOP operates at an I/O data transfer rate of 1866MB/s, with a 64-bit I/O bandwidth. The chip also features special heat-resistant properties” [20]. It is believed by Samsung that the new chip will “decrease the total area used by approximately 40%” [20]. Research shows that Apple began marketing its tablets “as a consumer device that would fill the gap between smartphones and laptop computers” [1]. Samsung’s focus has been different. In fact, “now that Samsung's ambition has grown to creating, using, and selling chips of its own, it makes sense to develop custom cores and achieve CPU design freedom. The company already had a flying start in custom CPU making, coming up with the impressive Exynos 7420 SoC, which powers the Samsung Galaxy S6 and S6 Edge. But this chipset employs licensed ARM Cortex-A57 and Cortex-A53 designs, although it is made on Samsung's own FinFET 14nm fabrication process. Hopefully, Samsung's CPU core design ventures will enjoy similar success. The company ‘expects to see results’ from the project by the first quarter of next year, and aims to use in-house processors for its high-end products. In addition, Samsung has been progressing with integrating mobile processors and modems on a single die, which could put its chipsets in the same league as Qualcomm's Snapdragon platform. Samsung started pretty late with the whole processor thing, but it seems eager to grow” [3].

The theoretical framework for the study is based on the theory of computation. This can be divided into different areas: complexity, computability, and automata [21]. Complexity theory focuses on the classification of “problems according to their ‘difficulty’” [21]. Computability theory focuses on the classification of “problems as being solvable or unsolvable” [21]. Automata theory focuses on the determination of whether or not “these models have the same power, or if one model can solve more problems than the other” [21]. For this study, the computability theory

is being used. This is because problems are classified in one of two ways. Since the purpose of the study is to determine how I/O, IRQ, and firmware drivers can best influence performance, the results of these components can be defined as “solvable” or “unsolvable.” However, it is hypothesized that the theory can be expanded to define the components as “efficient” or “inefficient.” This can lead to hypotheses and null hypotheses related to the study, which can then be analyzed utilizing the Chi Square Test.

The methodology is based on a statistical analysis. The chapter discusses the variables and measurements, as well as research design and analysis. In the conceptual framework, it is seen that I/O, IRQ, and firmware drivers interact to create a low cost, high performance tablet. Therefore, these are the primary variables and are used to determine whether or not the tablet would be efficient or inefficient. For this study, the I/O speeds are considered to be the data rates for the most common I/O devices and can be found in Tables 1 and 3, respectively. The devices being used are: keyboard, mouse, 56K modem, scanner, digital camcorder, 802.11g wireless, 52x CD-ROM, fast Ethernet, compact flash card, FireWire (IEEE 1394), USB 2.0, SONET OC-12 network, SCSI Ultra 2 disk, gigabit Ethernet, SATA disk drive, Ultrium tape, and PCI bus. These rates will be considered individually with the other two variables: IRQ and firmware drivers. For the purposes of this study, IRQ is determined through non-maskable interrupts (vectors 1–31). Finally, driver speed is considered to be 500 MB. This is because most computers need at least this amount to run necessary operations. This efficiency will be calculated by determining the product of the three averaged variables, dividing this number by a dummy variable, and finding the square root of the quotient.

The dummy variable comes from measuring efficiency in different CPU architectures. The architectures being measured include: OMAP3640, QSD8250, Tegra2, i.MX515, TCC8803, WM8650, SSPV210, RK2918, A10 / A13, and ATM709. Each of these architectures is measured for efficiency through performance and cost. The highest rate any architecture could obtain was 10, while the lowest was 2. Individually, each factor was scored on a scale between 1 and 5, with 5 being the highest score possible. This information will be utilized in a statistical analysis to compare the effects of each of the variables. This calculated efficiency rate will be

compared to the already established efficiency rate from Table 5. This will be used to conduct a Chi Square Test analysis to determine the rate of efficiency for the different variables. The Chi Square Test is important because it shows whether or not "a relationship observed in a sample that would have been unlikely to occur if really there is no relationship in the larger population" [22]. The most common significance is $p < 0.05$, so it will be used in this scenario as well. The efficiencies can be compared in a scatter graph and used to conduct a linear regression analysis. With this information, further relationships can be determined between the variables. This will include measures of central tendency in order to determine average efficiencies for each of the variables.

The results chapter is designed to show the efficiency of I/O, IRQ, and firmware drivers on the different architectures (identified as dummy variables N1 through N10).The first portion of the chapter converts the data rates of the different I/O devices to megabytes. This allowed the researcher to develop the values for variable A. The same activity was conducted for IRQ. The researcher used bytes. Since the entire table was worth bytes, the researcher elected to multiply each vector number (1 through 31) by 1024 bytes (the entire worth of the table) and convert it to megabytes to develop the values for variable B. In order for most CPUs to run efficiently, they need 500 MB of RAM free. This is being used as the firmware driver variable C. The dummy variable N, was found previously through measuring performance and cost for different CPU architectures. Since the dummy variable was used as a stabilizing influence, variables A and B were averaged. Thus, the average for variable A was 94.82044505. The average for variable B was 0.015625. It was determined that if the researcher elected to multiply each variable individually against the dummy variables, it would yield 5,270 results. Thus, by conducting averages, the researcher was able to obtain useful results in less space. Therefore, the efficiency test was conducted ten times, once for each of the dummy variables. While the established efficiencies ranged from 7 to 10, the tested efficiencies ranged from 8.6 to 10.3. The average for the established efficiency was 8.5. The average for the tested efficiencies was 9.4. This suggests tentatively that I/O, IRQ, and firmware drivers have different impacts on performance.

This was tested further using a Pearson's Correlation Coefficient test. This showed an r value of -0.99, which corresponded with the downward slope in the

efficiency in Figure 7, showing a negative correlation with a downward slope. This suggests as the efficiency of one variable increases, the efficiency of another variable decreases. This corresponds with the literature review. For instance, in the literature review, it was found that polling was not always as efficient as IRQ and vice versa.

The study wants to determine the influence of I/O, IRQ, and firmware drivers on efficiency. This can be done using hypotheses and null hypotheses, as well as the data already obtained. This study has used a dummy variable for a consistent variable and calculated a new efficiency variable for each of the architectures. Therefore, these are the two factors used to conduct the Chi Square Test. The first section involves establishing the hypothesis and null hypothesis. In this case, the hypothesis is: I/O, IRQ, and firmware drivers increase efficiency in CPUs. The null hypothesis is: I/O, IRQ, and firmware drivers do not increase efficiency in CPUs. Thus, with a significance of $p<0.05$, the critical value is 3.940. In this case, if the Chi Square is lower than the critical value, the null hypothesis is rejected. Since the Chi Square is 1.045 and is lower than the critical value of 3.940, the null hypothesis is rejected. Therefore, it is confirmed that I/O, IRQ, and firmware drivers increase efficiency.

CHAPTER V
CONCLUSION

The goal of this chapter is to discuss the benefits of the study, problems of the study, limitations of the study, and future recommendations. It is believed that through this discussion, it would be possible to develop more efficient tablets at a lower price, yet offer comparable products to those that are more expensive.

5.1 Benefits of Study

The study is beneficial in a variety of different ways. To begin with, since companies like Samsung and Apple are designing their own chips, they are able to create optimal efficiency for their microprocessors. This allows these companies to create low cost processors; thus, decreasing the cost of their tablets. This is highly beneficial to both the population and the companies. For example, one article showed that the price of the Apple iPad was too expensive for many consumers. Therefore, a low cost tablet could be highly beneficial.

For instance, in China, the tablet by Xiaomi costs 1499 to 1699 yuan, whereas the comparable iPad mini costs 2888 yuan. This shows that the Xiaomi tablet is approximately half the cost as that of the iPad mini. As a result, it can be concluded that a lower cost tablet would have increased sales volumes. Hypothetically, for every one iPad mini sold, two Xiaomi tablets could be sold. This means that for each 2888 yuan Apple earns, Xiaomi is earning 2998 yuan to 3398 yuan. If the tablet is comparable in quality and efficiency, this rate could be even higher. These differences in focus show that a cheaper tablet would be highly beneficial in the long run.

5.2 Problems of Study

The study is beneficial in a variety of different ways. To begin with, since companies like Samsung and Apple are designing their own chips, they are able to create optimal efficiency for their microprocessors. This allows these companies to create low cost processors; thus, decreasing the cost of their tablets. This is highly beneficial to both the population and the companies. For example, one article showed that the price of the Apple iPad was too expensive for many consumers. Therefore, a low cost tablet could be highly beneficial.

There are other ways to meet I/O needs. For instance, scheduling I/O can be used. Research shows that “scheduling I/O requests can greatly improve overall efficiency. Priorities can also play a part in request scheduling; buffering and caching can also help, and can allow for more flexible scheduling options” [21]. The research showed that “management of I/O devices is a very important part of the operating system so important and so varied that entire I/O subsystems are devoted to its operation. (Consider the range of devices on a modern computer, from mice, keyboards, disk drives, display adapters, USB devices, network connections, audio I/O, printers, special devices for the handicapped, and many special-purpose peripherals.) I/O Subsystems must contend with two conflicting trends: (1) The gravitation towards standard interfaces for a wide range of devices, making it easier to add newly developed devices to existing systems, and (2) the development of entirely new types of devices, for which the existing standard interfaces are not always easy to apply” [21]. One way to meet these differences is through buffering. In fact, buffering is done “(1) to account for speed differences between two devices. A slow device may write data into a buffer, and when the buffer is full, the entire buffer is sent to the fast device all at once. So that the slow device still has somewhere to write while this is going on, a second buffer is used, and the two buffers alternate as each becomes full. This is known as double buffering. Double buffering is often used in animated graphics, so that one screen image can be generated in a buffer while the other completed buffer is displayed on the screen. This prevents the user from ever seeing any half-finished screen images. (2) Data transfer size differences. Buffers are used in particular in networking systems to break messages up into smaller packets for transfer, and then for re-assembly at the receiving side. (3) To support copy semantics. For example, when an application makes

a request for a disk write, the data is copied from the user's memory area into a kernel buffer. Now the application can change their copy of the data, but the data which eventually gets written out to disk is the version of the data at the time the write request was made. Caching involves keeping a copy of data in a faster-access location than where the data is normally stored. Buffering and caching are very similar, except that a buffer may hold the only copy of a given data item, whereas a cache is just a duplicate copy of some other data stored elsewhere. Buffering and caching go hand-in-hand, and often the same storage space may be used for both purposes. For example, after a buffer is written to disk, then the copy in memory can be used as a cached copy, until that buffer is needed for other purposes" [21]. Other issues are caused by I/O. For example, "I/O requests can fail for many reasons, either transient (buffers overflow) or permanent (disk crash). I/O requests usually return an error bit (or more) indicating the problem. UNIX systems also set the global variable error to one of a hundred or so well-defined values to indicate the specific error that has occurred. Some devices, such as SCSI devices, are capable of providing much more detailed information about errors, and even keep an on-board error log that can be requested by the host. The I/O system must protect against either accidental or deliberate erroneous I/O. User applications are not allowed to perform I/O in user mode - All I/O requests are handled through system calls that must be performed in kernel mode. Memory mapped areas and I/O ports must be protected by the memory management system, but access to these areas cannot be totally denied to user programs. (Video games and some other applications need to be able to write directly to video memory for optimal performance for example.) Instead the memory protection system restricts access so that only one process at a time can access particular parts of memory, such as the portion of the screen memory corresponding to a particular window" [21].

5.3 Limitations of Study

Previous studies show that different types of CPU core architectures have been used to create various low cost tablets that are comparable to the iPad. Although this research has been effective in many cases, none have reached the specifications necessary to create the most comparable tablet.

The purpose of this study was to determine ways to develop a quality product at a minimum price. It was found that it is possible to develop a high performance and high quality product such as a tablet, provided there is ample time for researching the needs of consumers and availability of products. Furthermore, it is necessary to have ample financial support to the research process without focusing on making a profit.

The evidence obtained from the research underscored the assessment scores and provided insights that were not expected. Thus, navigation of future research and employed technology needs to be transparent. This is because technology issues do sometimes occur, creating user and developer frustration, which resulted in lower scores. For instance, when participants indicated frustration with the speed of the tablet or a lack of interactive multimedia, scores were generally lower. Thus, more research is necessary in order to incorporate the software, I/O device, and CPU core architecture to provide an optimal, stable, well formed, well designed, and an adaptable performance tablet environment.

5.4 Future Recommendations

Significantly, “the I/O system is a major factor in overall system performance, and can place heavy loads on other major components of the system. Interrupt handling can be relatively expensive (slow), which causes programmed I/O to be faster than interrupt-driven I/O when the time spent busy waiting is not excessive. Network traffic can also put a heavy load on the system. Other systems use front-end processors to off-load some of the work of I/O processing from the CPU. For example a terminal concentrator can multiplex with hundreds of terminals on a single port on a large computer. Several principles can be employed to increase the overall efficiency of I/O processing: reduce the number of context switches; reduce the number of times data must be copied, reduce interrupt frequency, using large transfers, buffering, and polling where appropriate; increase concurrency using DMA; move processing primitives into hardware, allowing their operation to be concurrent with CPU and bus operations; and balance CPU, memory, bus, and I/O operations, so a bottleneck in one does not idle all the others. The development of new I/O algorithms often follows a progression from

application level code to on-board hardware implementation, Lower level implementations are faster and more efficient, but higher level ones are more flexible and easier to modify. Hardware level functionality may also be harder for higher level authorities (e.g. the kernel) to control" [21].

As a result, it is recommended that further studies be conducted regarding I/O efficiency in different circumstances such as buffering, caching, DMA. This information would be valuable in creating a low cost tablet because it would assist in determining the best way to make these placements and what methods would be most useful. Furthermore, through the analysis of the different types of I/O placement and techniques used, it may be possible to create a tablet that exceeds the more expensive tablets, such as the iPad mini and Samsung Galaxy, at an even lower rate. Information like this can assist companies in meeting their strategic and profit objectives at a lower cost, yet exceed consumer expectations through the creation of a highly efficient tablet.

REFERENCES

[1] Statistica, “Apple iPad global sales 2010-2015, by quarter | Statistic,” 2015. [Online]. Available: http://www.statista.com/statistics/269915/global-apple-ipad-sales-since-q3-2010/. [Accessed: 08-May-2015].

[2] PC Pro, “The 17 best tablet of 2015: what’s the best tablet to buy right now? | PC Pro,” 2015. [Online]. Available: http://www.pcpro.co.uk/tablets/7021/the-17-best-tablet-of-2015-what-s-the-best-tablet-to-buy-right-now#besttablets. [Accessed: 08-May-2015].

[3] L. D, “Samsung turns to designing custom CPU cores for its Exynos mobile chipsets,” 2015. [Online].Available : http://www.phonearena.com/news /Samsung-turns-to- designing -custom-CPU-cores-for-its-Exynos-mobile-chipsets_id67345. [Accessed: 09-May-2015].

[4] K. Wagner, “Apple Hires Away Top Samsung CPU Designer to Build Custom Apple Chips,” 2012. [Online]. Available: http://gizmodo.com/5951051/ apple-hires-away-top-samsung-chip-designer-to-build-custom-apple-chips. [Accessed: 09-May-2015].

[5] J. Kaustubh, “I/O Subsystems.” Columbia University, 2013.

[6] L. Wong, “Programmed I/O, Interrupt & Direct Memory Access (DMA),” 2009. [Online]. Available: http://www.louiewong.com/archives/137. [Accessed: 09-May-2015].

[7] “5.7. Input-Output and Interrupt.” [Online]. Available: http://www.cs.uwm.edu/ classes/cs458/Lecture/HTML/ch05s07.html. [Accessed: 09-May-2015].

[8] R. Fryar, “I/O Requests Taking Longer Than 15 Seconds To Complete...” [Online]. Available: http://www.sql-server-pro.com/i-o-requests-taking-longer-than-15-seconds-to-complete.html. [Accessed: 09-May-2015].

[9] A. S. Tanenbaum, Modern Operating Systems, 3rd ed. Prentice-Hall, Inc., 2008.

[10] H.-M. Chen, P.-H. Chen, T.-J. Pan, and F. Lai, “Designing Platform-based System Power Management on a Smart Tablet Appliance,” IEEE, 2003.

[11] L. Gwennap, “MIPS R10000 Uses Decoupled Architecture,” MicroDesign Resour., 1994.

[12] L. Gwennap, “MIPS R12000 to Hit 300 MHz,” MicroDesign Resour., 1997.

[13] P. Mudliar and J. Pal, “ICTD in the Popular Press: Media Discourse Around Aakash, the ‘World’s Cheapest Tablet,’” Inf. Technol. Int. Dev., vol. 11, no. 1, pp. 41–55, 2015.

[14] Y. Yang, “The Taiwanese Notebook Computer Production Network in China: Implication for Upgrading of the Chinese Electronics Industry,” 2006.

[15] M. Kan, “China’s Xiaomi breaks into tablet market, launches low-price 4K TV,” PC World, 2014. [Online]. Available: http://www.pcworld.com/article/2155580/chinas-xiaomi-breaks-into-tablet-market-launches-lowprice-4k-tv.html. [Accessed: 09-May-2015].

[16] B. Powell, “Can Apple win over China?,” Fortune, 2012. [Online]. Available: http://fortune.com/2012/10/11/can-apple-win-over-china/. [Accessed: 09-May-2015].

[17] Y. J. Yu, D. I. Shin, W. Shin, N. Y. Song, J. W. Choi, H. S. Kim, H. Eom, and H. Y. Yeom, “Optimizing the Block I/O Subsystem for Fast Storage Device,” ACM Trans. Comput. Syst., vol. 32, no. 2, 2014.

[18] P. Garrou, “Battling over Apple,” Solid State Technol., p. 11, 2015.

[19] A. Maheshwari and M. Smid, Introduction to Theory of Computation. 2014.

[20] Penn State Science, “Statistical Significance of Observed Relationship / Chi-Square Test,” 2015. [Online]. Available: https://onlinecourses.science.psu.edu/stat200/node/73. [Accessed: 10-May-2015].

[21] “Operating Systems: I/O Systems.” [Online]. Available: http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/13_IOSystems.html. [Accessed: 10-May-2015].

BIOGRAPHY

NAME	Mr. Yodchai Singhsathitsukh
DATE OF BIRTH	5 June 1970
PLACE OF BIRTH	Songkhla, Thailand
INSTITUTIONS ATTENDED	University of Kwazulu Natal, 1991-1995 Bachelor of Science (Computer Science) Mahidol University, 2013-2015 Master of Science (Information Technology Management**)**
HOME ADDRESS	69/111 M.10. M. Ban Home Place Wongwan, Khlong Thanon Road, Rattanatibet, Bangmaenang, Bangyai Nonthaburi, 11140 Tel No: +66-89-8946256 Email: charles@keysoft.co.th