Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

DESIGN, IMPLEMENTATION, AND EVALUATION OF A SHARED-MEMORY PARALLEL PROCESSING SYSTEM (SMPPS)

by
Eric H. Staub

As technology reaches its limits of improvements in microprocessor processing speeds, scientists and engineers have to find viable solutions to meet ever-increasing demands for faster processing speed. One such solution is parallel processing. No longer does one have to wait on sequential operations. A specific task can be split in sub-tasks that can run simultaneously, thus reducing the overall execution time of the task.

The design and implementation of these systems is crucial to the effectiveness of parallel systems. A dual-processor SMPPS was designed and implemented in order to demonstrate how multiple processors are a viable solution to increasing the speed of computer processing. Parallel algorithms were developed for this system and were used for performance analysis. The results show that SMPPS systems of a small scale can result in very significant increases in speed for problems characterized by fine-grain parallelism.

DESIGN, IMPLEMENTATION, AND EVALUATION OF A SHARED-MEMORY PARALLEL PROCESSING SYSTEM (SMPPS)

by
Eric H. Staub

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Department of Electrical and Computer Engineering

January 1999

APPROVAL PAGE

DESIGN, IMPLEMENTATION, AND EVALUATION OF A SHARED-MEMORY PARALLEL PROCESSING SYSTEM (SMPPS)

Eric H. Staub

Dr. Sotirios G. Ziavras, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, and Computer and Information Science, NJIT

Dr. Solomon Rosenstark, Thesis Co-Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Edwin Hou, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Eric H. Staub

Degree: Master of Science in Computer Engineering

Date: January 1999

Undergraduate and Graduate Education:

- Master of Science in Computer Engineering
  New Jersey Institute of Technology, Newark, NJ, 1998

- Bachelor of Science in Computer Engineering
  New Jersey Institute of Technology, Newark, NJ, 1997

Major: Computer Engineering

ACKNOWLEDGMENT

The author wishes to express sincere thanks to my two advisors, Dr. Ziavras and Dr. Rosenstark, for their guidance, support, knowledge, and mentoring. I would also like to thank Dr. Hou for serving as a committee member.

I would like to give a special thanks to Rosalie Gaddala for her friendship and support throughout my academic career at NJIT and to Amy Sun who was an inspiration for me over the last few months of my graduate work.

Most of all, I would like to thank my Mother and the rest of my Family for the love and support that has gotten me this far.

I would like to thank all of the Electrical and Computer Engineering Department and all of the Faculty and Staff of NJIT that have been an integral part my academic career at NJIT. I would like to thank the numerous students that I had the opportunity to work and study with. I would like to thank the United States Air Force for giving me the opportunity to become an officer to serve my country and the Air Force Institute of Technology for giving me the opportunity to get my graduate degree at NJIT. I would like to thank the AFROTC Detachment 490 Staff, past and present, for their guidance and support while I have been at NJIT. Also, I would like to thank my brothers of Tau Delta Phi Fraternity. And finally, I would like to thank Altera's University Program for supplying hardware and software for this project.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

Figure **Page**

LIST OF DIAGRAMS

CHAPTER 1

INTRODUCTION

1.1 Parallel Processing

1.1.1 Importance of Parallel Processing

Even with ever changing technology, industry is always looking for ways to improve performance. Scientists are continually finding innovative ways to speed up the processing power of computers. Still, we need faster and more effective ways to accomplish a task. Now that advancements in technology are reaching their limits, industry must look for a new way to keep up with the demands. There is the old adage that two minds are greater than one. This theory can be applied to computer processing. With two processors, not only can more tasks be accomplished, but also tasks can be accomplished faster.

For example, the simple task of {g = (a+b)*(c+d)} would take three steps (part a. of **Figure 1**) on one computer. On a system with two processors, that same task would take two steps (part b. of **Figure 1**). For simplicity sake, the time to pass information between the processors is not considered.

(a)System with one processor.

Step 1: Processor A adds 'a' to 'b' and places value in 'e'.
Step 2: Processor A adds 'c' to 'd' and places value in 'f'.
Step 3: Processor A multiplies 'e' and 'f', and places in 'g'.

(b)System with two processors.

Step 1: Processor A adds 'a' to 'b' and places value in 'e'.
Processor B adds 'c' to 'd' and places value in 'f'.
Step 2: Processor A or B multiplies 'e' and 'f', and places in 'g'.

Figure 1: Steps processors make to solve the equation g = (a+b)*(c+d).

This is a 33% improvement in the time to accomplish a simple task. If the additional processor gives a 33% increase, why not add another processor? In this simple case the addition of more processors would not have any effect. This is because the task is made up of three subtasks, one of which requires information from the previous two. Even if the third processor was assigned the multiplication of 'e' and 'f' it would not be able to proceed until the additions were complete.

One might conclude that the improvement of processing time using multiple processors is limited. Actually the limit only exists for a particular task. As the task changes, the speedup factor changes. When multiple processor theory is applied to the task of (a+b)*(c+d)*(e+f)*(g*h), the results are quite different. On one processor the task will take seven steps. On a two-processor system it would take four steps. This is over 40% decrease in processing time. On a four-processor system that same task would take only three steps. This is over 50% decrease. If the task is applied to a five-processor system, there is no improvement in processing time. Once again the processing time can only be improved to a certain limit.

Another factor to consider is that adding a fourth processor only increased the speedup by 10%. When one processor was added there was a gain of 40%, and only 10% more when adding two additional processors. Also, during some of the steps, some of the processors are not needed. Further complicating the matter is the movement of data between processors. This transfer will take additional time that will decrease the overall speedup of the system. Deciding what is the best possible design to obtain the best possible results is a topic that will not be discussed in detail and will be left to independent research. However, the focus of this paper will center on the design of a shared-memory parallel dual-processor system and the timing results of running algorithms on the system.

1.1.2 Classes of Parallel Processing

Before I get into the design of the system, I will discuss the different types of parallel computing systems. As one might guess, parallel systems are designed in different ways. In general, parallel systems are classified in to two major groups. The system I have designed falls into the shared-memory class and the other class consists of message passing systems. Each system has its pros and cons and the type of system needed is basically dependent on the task that needs to be accomplished. How parallel computers communicate with one another and how they share memory determines which one of the two major classes of parallel computers the systems belong to.

Systems that are considered inherent parallel computers are those which operate in the MIMD (multiple instruction stream over multiple data stream) mode. An example of a MIMD system is shown in **Figure 2.** Since parallel computers must share information, there has to be a way for them to access the shared information. In multiprocessor shared-memory systems this is accomplished by placing information in some variable and giving all systems access to that variable. In message-passing systems the information is passed between computers by using an interprocessor communication network.

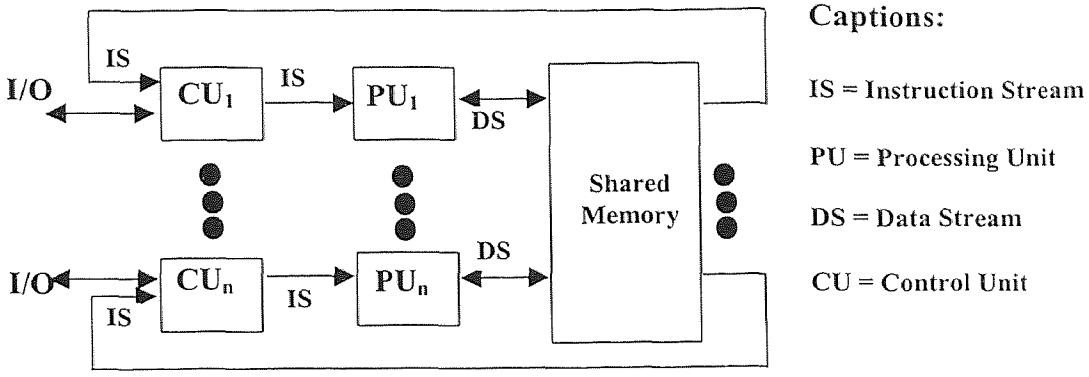


Figure 2: MIMD architecture (with shared-memory).

1.2 Existing Machines

1.2.1 Message-Passing

A system in the message-passing class consists of one or more multiple-computer networks. These networks connect together computer nodes. The computer nodes communicate information between one another through these networks. Hardware routers usually handle this communication. An example of a message-passing interconnection network is shown in **Figure 3.**

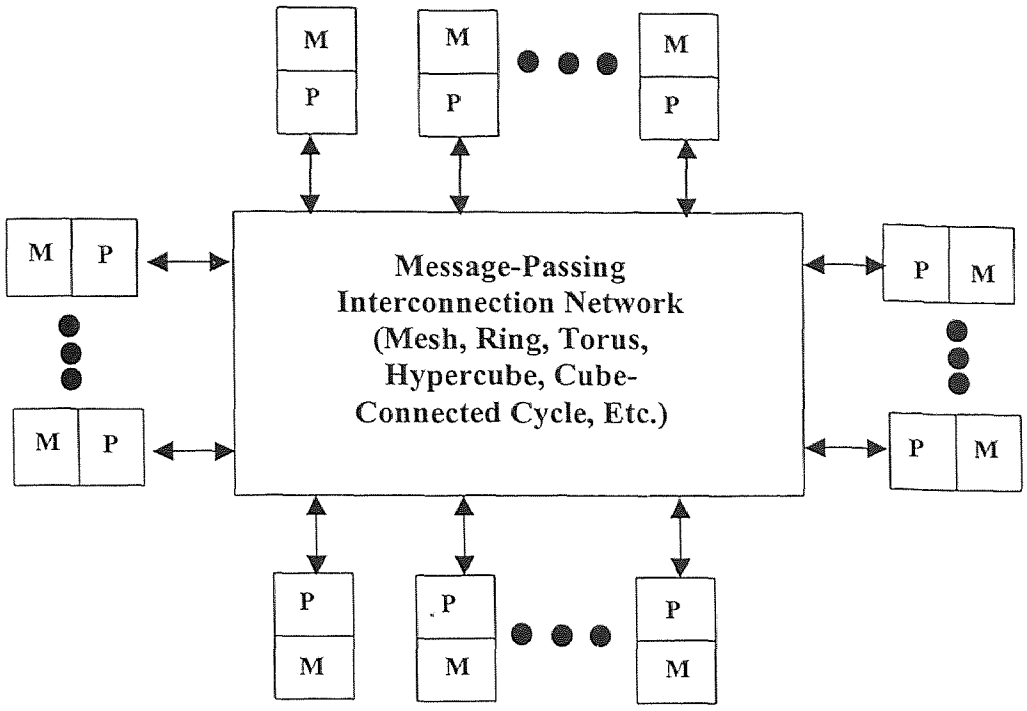


Figure 3: Generic model of a message-passing multicomputer (M=Memory, P=Processor).

Each network node is attached to a router. Based on the design and type of protocols that the router uses, information is then sent between the computer nodes via routing. This gives the designer the flexibility of creating multiple types of communications between the networks. By changing how the networks interact, the designer has the ability to use the same networks to accomplish numerous different tasks.

As with all technology, the scientist and engineer strive to improve the original design. Message-passing systems are now in their third stage of development. Development started in 1983 with systems like the Caltech Cosmic and the Intel iPSC/1. These systems were designed with software-controlled message-passing for the hypercube architecture.

Over the years of 1988-1992, systems such as the Intel Paragon and the Parsys SuperNode 1000 represented the next stage in the evolution of message-passing systems. The systems incorporated routing messages via hardware, utilizing software for medium-grain distributed computing, and using mesh-connected architectures.

The third stage of the development started in 1993 and consisted of machines that placed processing and communication devices on the same chip. Systems such as the MIT J-Machine and the Caltech Mosaic are based on this design.

Listed above are a few of the many systems that have been developed. Each system has its own unique design. What that design is and how each accomplishes its message passing can be found in numerous technical notes and publications. These systems were mentioned just to give a flavor of the type of systems and progression of the development of message-passing systems.

1.2.2 Shared-Memory

Shared-memory systems consist of multiple-processors, each of which has its own private memory, and information is shared through an independent memory that all of the processors have the ability to access. As with message-passing systems, I will give a brief description of shared-memory systems. I will briefly describe only three of the many models of shared-memory systems. Many other models incorporate one or more features of these three models.

The first model, **Figure 4,** is the uniform-memory-access (UMA). In this model all processors have equal access to all memory. These systems are for multiple processes for problems characterized by a high degree (that is fine-grain) parallelism. The system I designed falls under this model.

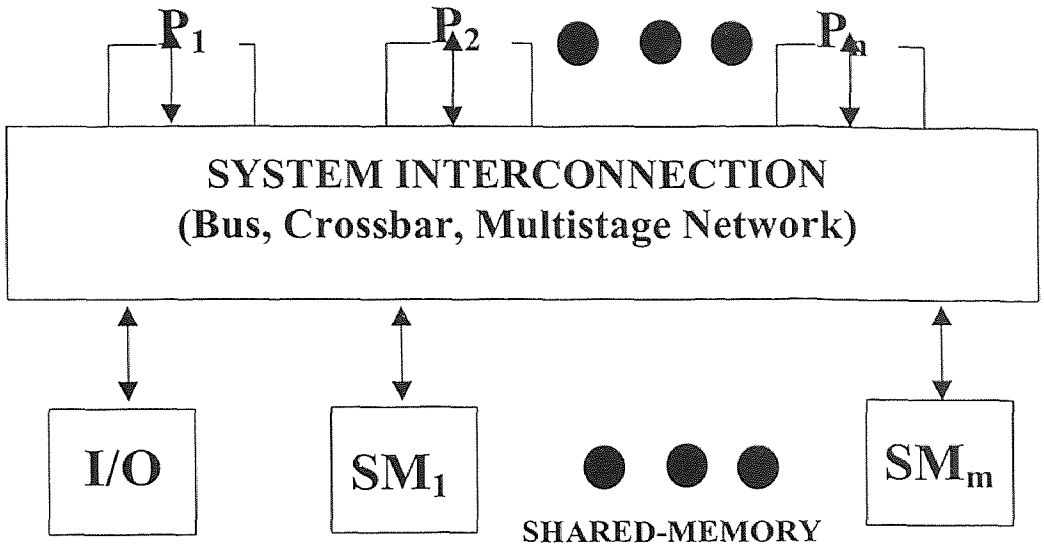


Figure 4: The UMA multiprocessor model (e.g., the Sequent Symmetry S-81)
[P = Processor; SM = Shared-Memory; I/O = Input/Output].

The next model, **Figure 5,** is the non-uniform-memory-access (NUMA). NUMA systems consist of groups of multiple-processors that are connected by interconnection networks. There is local-shared-memory within each group and global-shared-memory between the groups. These systems share memory based on the location of the memory in relation to the processor needing access to that memory. Therefore, the access time to memory is not uniformly distributed among the processors.

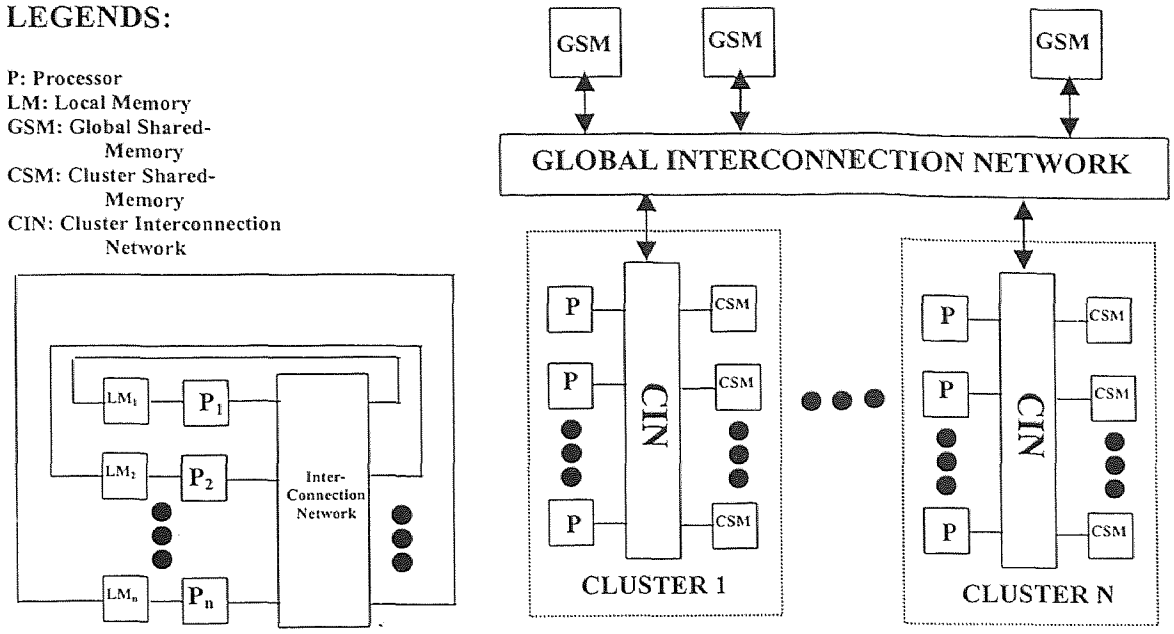


Figure 5: Two NUMA models for multiprocessor systems.

The last model, **Figure 6**, I will discuss, is the cache-only memory access (COMA). These systems are similar to NUMA systems, but the shared memories are replaced with cache memories. Processors wanting to access memory in another processor's cache memory must do so through cache directories.

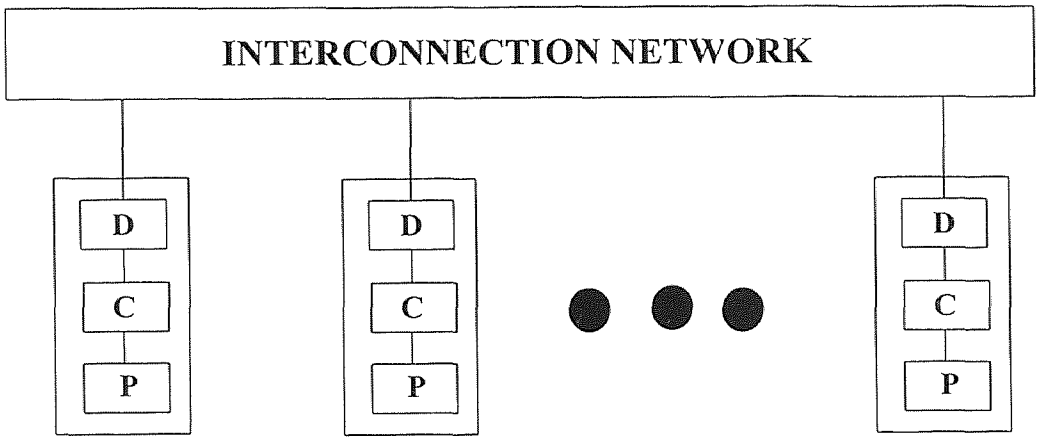


Figure 6: The COMA model of a multiprocessor (D: Directory, C: Cache, P: Processor; e.g., the KSR-1).

Numerous different sources, including the Internet, can be found for further information about parallel systems. This follow-on information is not necessarily needed to understand the design of my shared-memory system or the results of testing algorithms on that system.

CHAPTER 2

IMPLEMENTING A SHARED-MEMORY PARALLEL PROCESSING SYSTEM (SMPPS)

2.1 Objectives

There are three main objectives to this project. The first is the design of the shared-memory parallel processing system. Next is the implementation of that system. The final objective is the evaluation of the system for some algorithms.

2.2 A Dual-Processor Shared-Memory Parallel Processing System

2.2.1 Meeting Design Objectives

Since the evaluation of the system consisted of testing algorithms, I needed to design a system that could be implemented within time and monetary constraints. This system would have to show the effectiveness of running an algorithm on a parallel system as opposed to running that same algorithm on a single processor system.

I chose to develop a system with two processors and a single shared-memory. This would reduce the cost and complexity of the project. Also, it would help keep me within the time and monetary constraints. The next step was to determine which processor to use for the project.

I initially chose to use the TI TMS320C80 processor. The C80 processor consists of four DSPs and one RISC processor. I spent the next month gathering information about the C80. I considered how I would implement a system using two C80 processors

and what software would have to be developed to manage and test the interface between the two processors. After carefully considering the options that the information I collected presented to me, I determined that I would be unable to use the C80 for this project. Using the C80 would not only be cost prohibitive, but the complexity of implementing a dual processor system was extremely complex.

I then focused my attention on using TI's C40. Even though the cost was quite less, the complexity still remained quite high. After another month of investigations it was determined that using the C40 was not a viable solution. This left the Motorola 68000 series microprocessor. These processors would be much more cost effective and the complexity would be greatly reduced. Since I was familiar with this series of microprocessor, I determined that it would be the most promising candidate for a dual-processor system.

2.2.2 The Design

As an undergraduate, I was involved in many projects. The most significant was my senior project. In this project I developed a control system for a constant-pressure floodgate. I used the Motorola 68008 microprocessor as the control system processor. I used a micro-controller design that was developed by Dr. Rosenstark and is part of EE-393, Electrical Engineering Lab III. The micro-controller design and specifications are explained in detail in the EE-393 Lab Manual.[(Rosenstark 1998)] The current version of the Lab Manual has the new micro-controller, Motorola 68EC000 microprocessor, in place of the Motorola 68008 microprocessor.

Once it was determined what microprocessor I should be using, the project was set in motion. The Electrical Engineering Laboratory III (EE 393 – Spring 98) was using the last of the MC68008 to build micro-controllers. Since the discontinuation of the processor, Dr. Rosenstark was seeking an alternative processor. The alternative was the MC68EC000. To test the feasibility of using this processor, Dr Rosenstark had one student build a micro-controller with the MC68EC000. The student was successful in using MC68EC000.

In order to accomplish the objectives I set, I needed to make modifications to the micro-controller in the EE-393 Lab Manual. The micro-controller has its own memory, which included DRAM and an EEPROM. The memory used in the EE-393 Lab Manual was 28C64 EEPROM and 6264 DRAM. Since my design required a larger memory space, I chose to use an ATMEL 28C256 EEPROM and a 62256 DRAM. This would give me two blocks, each 8K bytes, of addressable memory. This change in address space changed the addressing scheme of the micro-controller (see **Figure** 7).

	28C64/6264	28C256/62256
EEPROM	0000 - 1FFF	0000 0000 – 0000 7FFF
Private Memory	2000 - 3FFF	0000 8000 – 0000 FFFF
Shared Memory	N/A	0001 0000 – 0001 7FFF
Parallel Port	6000	0001 8000
Serial Port	4000	0002 0000
	All values are in HEX	

Figure 7: Address location of devices.

Another benefit of using these chips is that they are 28-pin packages. This would allow me to use the original design while only changing two wires for each chip. The additional wires are address lines A13 and A14. These lines will be connected to pins that where originally no-connect pins on the EEPROM and will replace the nCE2 pin and a no-connect pin of the DRAM. This is shown in **Figure 8** and **Figure 9.**

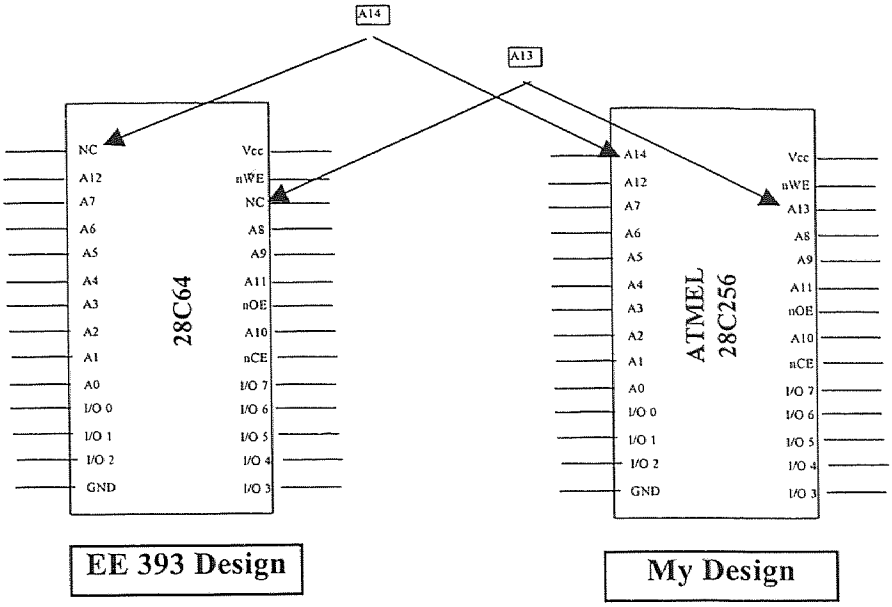


Figure 8: Differences between the 28C64 and the Atmel 28C256.

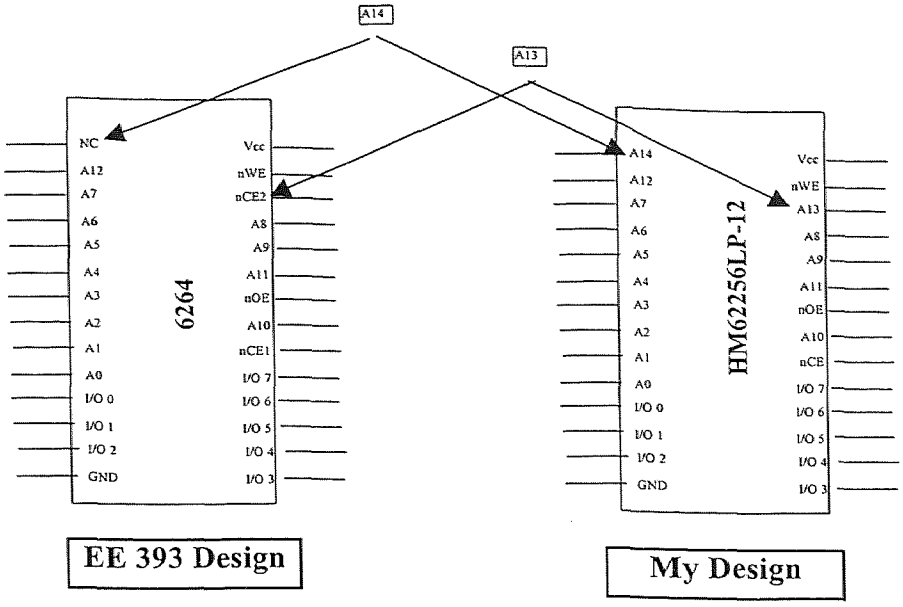


Figure 9: Differences between the 6264 and the HM62256LP-12.

Since I am using a larger address space, the address lines on the 74LS138 will have to change. Lines A13, A14, and A15 will be replaced with A15, A16, and A17 as shown in **Figure 10**.

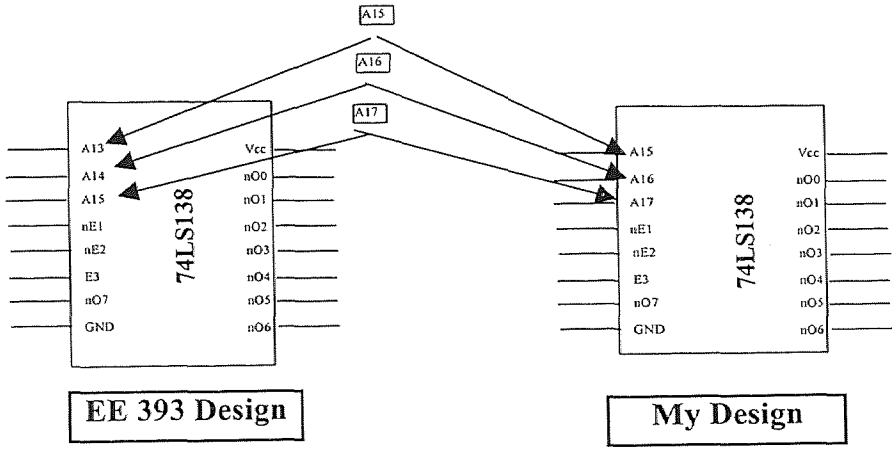


Figure 10: Differences in the wiring of the 74LS138.

Now that the major design decisions were out of the way I started to build the circuits around the microprocessor. I proceeded as far as possible with the parts that I had acquired up to this point. I was having difficulties acquiring some of the important components so I was unable to go any further. Due to lack of parts to complete the microprocessors I decided to work on the control logic and the 2-1 Mux.

After spending some time designing the control logic I received most of the components needed to finish the micro-controllers. After completing the first micro-controller, I ran into difficulties interfacing with the computer. Since I was only having trouble with communicating with the computer I started to build the second micro-controller. Once I completed this micro-controller, I ran into the same difficulties. After an exhaustive trouble shooting effort, I was only able to communicate with the computer on a simple level. I was still unable to run the Monitor program. I then changed my focus to the software and the assembler.

After more intense trouble shooting, Dr. Rosenstark and I determined that one of the problems was created by my larger address space. Specifically the range from 8000H to FFFFH. This problem was caused by the assembler when it sign extended. As a solution we decided not to use this address range. I moved the private memory to 0001 0000H – 0001 7000H and moved the shared-memory to 0002 0000H – 0002 7FFF. This solved some of the problems but I was still unable to get the monitor program to work.

While working on my project I was teaching EE393 over the second summer session. These students were using the MC68EC000. These students were using the smaller EPROMs and RAMs. They did not have the communication problems that I was having. This was very perplexing since it was the same program, except for the different

address scheme. Since I was able to communicate on a simple level it had to be a software problem. After using some unique debugging, I determined that the James L. Antonakos' Assembler was assembling addresses that used the LEA command with an offset of 6H. I also found another problem. The James L. Antonakos' assembler creates S1 records. This would not allow me to write a program to be loaded by the monitor in my memory location since my addressing scheme was a long word.

At this point I tried using another assembler. I found that Paragon's assembler was able to assemble the program, and I was able to run the monitor program. This created another problem. The Paragon assembler used S2 records in the Hex file. The monitor was not able to load S2 files, so I would not be able to load a program into memory.

Working with Dr. Rosenstark we came of with several solutions. The first was to change the LEA commands to MOVEA.L commands. This solved most of the problems but I would still be unable to use Antonakos' assembler for files to be loaded into the memory because my addressing scheme requires S2 records. Dr. Rosenstark's changing the monitor program to load S2 records solved this problem. Dr. Rosenstark has passed this information on to James L. Antonakos and he is currently working on a solution.

I now had two fully working micro-controllers. Now it was time to start to work on the shared-memory logic. For simplicity, I chose to make the shared-memory the same type as the private-memory of the micro-controllers. This way I would be able to use the same address and data bus as the micro-controllers.

The next step was to design the interface between the micro-controllers and the shared-memory. My design called for single-port access of the memory. Also, access of the shared-memory should not interfere with the independent processing of the other processor unless both processors try to access the shared-memory at the same time. In order to accomplish that, I needed to separate the address and data buses of the individual processor while allowing access to those buses when shared-memory is accessed.

Diagram I in Appendix A shows the initial block diagram for the system. I separated the address buses with 2-1 multiplexors and the data buses with bus-transceivers. I used a bus-transceiver on the data bus because of the bi-directional nature of the data bus. After further evaluation of my design I found that I had unnecessary logic.

Diagram II in Appendix A shows that I removed two bus-transceiver blocks and two 2-1 MUX blocks. The DRAM chip has an enable pin on it. This enable pin would only be activated when a processor requires access to the shared-memory. This allowed me to remove the MUX blocks. The bus-transceiver is bi-directional so it can be placed in the direction of the shared memory while a processor is accessing its private memory. Since the shared-memory is not enabled during this time, the data on the data lines of the shared-memory chip is ignored. This allowed me to remove the bus-transceiver blocks.

Now that the design for the address and data bus was complete I needed to design the shared-memory control logic. The problem that needed to be solved was how to access the shared-memory with interrupting independent processing of the other processor. I used one of the features of the MC68EC000 to build my design.

I used the MC68EC000 A/S pin and the /DTACK pin. When an instruction is executed the MC68 places a signal on the A/S pin. In order for the processor to continue to the next instruction, a signal must be placed on the /DTACK pin. Once the state on the /DTACK pin has gone from high to low and then back to high, the processor will continue on to the next instruction. If the transition is not completed the processor will not continue.

Since my design requires that a second processor wait till the first processor is done when both processors try to access shared-memory, I can use these pins to my advantage. In the EE 393 design the two pins are connected directly together. If I could separate the pins during shared-memory access, I would have solved my problem. Now that I had a possible solution to this problem, I had to consider the other chips that needed to be controlled by this logic.

The shared-memory had to be enabled when accessed and whether the operation is a read or write must be handled. The bus-transceiver on the data bus must be enabled and the direction set. And finally the multiplexor on the address bus must be set correctly. This design would require large amounts of logic and testing would become a nightmare. Luckily, as part of my undergraduate work I used a software package by Altera called MAX+plus II.

I decided to use ALTERA programmable chips for the control logic and the 2-1 Mux. Using the Altera chips would be much more cost effective and would reduce the area required for the shared-memory system. Also these chips would allow flexibility in the design of the logic. The design could be easily modified and reprogrammed onto the chip.

MAX+plus II can be used to design entire logic devices from those as simple a gate to those as advanced as microcomputers. The designs can be created in text format or in graphical format. Once the design is complete, it can be thoroughly tested. If it does not meet the specifications needed, then it can be easily changed and tested again. This eliminates the need to build the circuits, test them, and then throw them away because they did not meet the specifications you had planned. Another advantage was that the design could be placed on a single chip the size of a computer processor. Not only would I save time and money, but also the space I needed for my control logic would be reduced.

Diagram III in Appendix A shows one of the preliminary designs. The final design for the most part was similar to this design. One of the features of MAX+plus II that is very useful is the ability to create default symbols. This allows the use of the same sub-design in multiple places. This became particularly useful when testing a specific point of the design.

I used this feature in two places in my design. One place was the point that became the focal point of fault with my original design. This will be explained as I describe the final design of the control logic. The second place is the 1-2 de-multiplexor I created. I would have had to create a third default symbol, but this symbol had already been created. This was the 2-1 mulitplexor.

The 1-2 DEMUX is shown in Diagram IV in Appendix A. I created it using tri-state buffers. This design allows one signal to be sent over a different line based on what is selected by the select pin. The drawback to this design is the high 'Z' output that is created when a line is not selected. This would be a problem when a processor is

working with its own memory. Then the input to the shared-memory logic would be high 'Z'. Since my design of this control logic requires a high or low signal to be present, I had to come up with another solution.

The simple solution was an open-collector buffer. Since the chip that I will be placing the design on does not support open-collector buffers in the design, I chose to route the 1-2 DEMUX output out of the chip and then back into the chip via an input pin. The signal would then go through the open-collector buffer and then back into the design on the chip. This would require an additional chip. Since I had saved large amounts of space by using the Altera chip, I didn't mind adding one additional chip.

In order to save additional space, I chose to design the 2-1 multiplexors for the address bus with the MAX+plus II software. Diagram V in Appendix A shows this design. This would require the use of two Altera MAX EPM7128SLC84-7 chips. Using two MAX chips still required less space than using 2-1 multiplexor chips. After running the control design through many simulations, I programmed the design into the second MAX chip. I then proceeded to wire the chip into the micro-controller. Before I could actually test the design, I had to wire the bus-transceivers for the data bus and the second MAX chip, which has the 2-1 Multiplexors for the address bus.

Once the wiring was complete, I started testing the design. The design did not work the way it was expected to. After days of testing and troubleshooting, I narrowed the problem down to a specific area in the design. I removed this area from the design and created a default symbol for this area. It is shown as default symbol 'ctest' in Diagram VI in Appendix A. This would allow me to redesign and test the problem area of the design.

After many days of testing and modifications, I determined that I would have to redesign this portion of the control logic. Any modifications I made to the design would either introduce a race condition into the logic or give total control of the shared-memory to one processor. Just before starting from scratch, I asked Scott Margo, an NJIT Electrical Engineering Ph.D. student what he thought might solve the problem. After evaluating the design, he came to the same conclusion that I should start over from the truth tables. The resulting truth table is shown in **Figure 11**.

Ain	Bin	Aout	Bout	A'out	B'out
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	Invalid	Invalid
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	Invalid	Invalid
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	Invalid	Invalid
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	Invalid	Invalid

Figure 11: Truth table for the shared-memory control logic.

Using the Karnaugh Maps in **Figure 12 (a)** and **(b)** the following equations emerged:

A'out = (Ain /Bin)+(Ain /Bout)+(/Ain Bin /Aout /Bout)

B'out = (/Ain Bin)+(Bin /Aout Bout)

A'out	00	01	11	10
00	0	0	1	1
01	0	0	0	1
11	0	0	0	1
10	0	0	1	1

(a)

B'out	00	01	11	10
00	0	1	0	0
01	0	1	1	0
11	0	1	0	0
10	0	1	0	0

(b)

Figure 12: Karnaugh Maps for the shared-memory control logic.

The resulting logic is shown in Diagram VII in Appendix A.

I tested this design by running it through several simulations. The results of these simulations were very promising. After compiling the control design with this new design, I programmed it into the MAX chip. This began the testing phase of the new control logic. I used the monitor program on each micro-controller to manually access the shared-memory. I was able to edit and display the shared-memory from both micro-controllers. This confirmed that the hardware design was complete.

The next step was to write a program that used software semaphores to lock the shared-memory. The program I wrote is in Appendix B. The program ran flawlessly on both processors. Not only did the hardware design work, but also the software-controlled locks were executing properly.

2.2.3 Timer Configuration

Before I could move on to the algorithms, I had to decide how I would track the execution times. The most effective way is to interface directly with the micro-controllers. This would allow the software to directly control the timer. Not only would this be more efficient, but it would also produce more accurate times.

I chose the Intel 8253-5 programmable interval timer to accomplish the task of timing the execution of the algorithms. The 8253 timer is a 24-pin dual in-line package with three 16-bit counters, each with a count rate of up to 2 MHz. The timer has five different modes of operation and four different ways of obtaining count values. I will be using mode 0, interrupt on terminal count, and will use 'Read/Load least significant byte first, then most significant byte' for obtaining the count value. The timer counts down from 2^{16}-1. This produces a 16-bit number.

The timer has an eight-bit data bus that can be easily interfaced with the micro-controller's eight-bit data bus. This data bus is used to read the count value in the count register. As stated before this is done with two reads of the chip. The first read is stored in one register and the second read is stored in another register. The final result is the combination of the two values, which is a 16-bit number.

Once I completed the interface of the chip to the micro-controller, I conducted preliminary tests on the timer chip. These tests were done to ensure the timer was working properly. Even though I chose to operate the timers at 1.2 MHz, I noticed that the timer was counting completely down several times. I was getting valid count values but had no way of telling how many times the counter started over. This could cause a problem when determining the speed up of the algorithms that I would be testing on the project.

In order to solve this problem I had to find a way to track how many times the counter reaches zero. This was one of the main reasons I chose to operate the timer in mode 0. In mode 0 the timer would count down to zero, and once zero was reached a high signal would be placed on the out1 pin of the timer chip. Now I had a way to keep

track of how many times the timer reached zero. Of course, it was not as simple as I thought.

Once the timer reached zero, the signal would be placed on the out1 pin. The timer would then continue to count down again. The problem with this is that the signal on the out1 pin was not reset. The only way to reset the out1 pin was to reset the entire timer and then restart the timer. This presented another problem. All of these actions would take time. Even though it was a very small amount of time, it was still enough to reduce the accuracy of the execution times of the algorithms.

The solution to this problem brought about the final design for the interface of the timer. Since the resetting of the timer would take time, I needed to halt the execution of the algorithm while I was resetting the timer. I accomplished this by using the external interrupts on the MC68EC000.

Using the 68's interrupts I could reset the timer and count the number of times the timer reached zero. This was accomplished by adding an interrupt service routine to the monitor program. The routine, which is written in assembly, is shown in **Figure 13**. Using the interrupts also required some additional hardware design.

```
      ORG     $6300

; This is Interrupt #4 Service Routine

      move.b  #$44,($18000)
      addi.l  #$01,(ICNT)      ; # times counter counts down
      move.b  #$30,(LCW)       ; Initializes the counter to mode 0
      move.b  #$00,(WC1LB)     ; Loads the count value
      move.b  #$00,(WC1MB)
      RTE
```

Figure 13: Timer Interrupt Service Routine (written in assembly).

While designing the hardware interface between the timer and micro-controller, I developed a way to totally automate the resetting of the timer and the reading of the final count value. This would require additional interrupts and logic for the interface. After a few weeks of testing designs, I decided just to use the interrupt for the resetting of the timer and keeping track of how many times the timer reached zero. I made this decision based on the fact that these additional features of automation were not really necessary and the fact that I would not be able to work out the bugs in the design in the time allocated for the timer design.

Since I was not using automation for the stopping and reading of the timer, I had to create a design that would allow the software to stop and read the timer. In the micro-controller design the 74LS138 is used to select different chips. This is accomplished by having three upper address lines connected to the 74LS138. By executing a read/write at the address location specified by the address lines that are connected to the 74LS138, a particular chip will be enabled. Since I was not using all of the locations available on the 74LS138, I decided to use it to help with the stopping and reading of the timer.

Now that my new design for the timer required additional logic, I decided to use the MAX+plus II software. I designed the logic and then added it to the design for the address bus multiplexors. This is shown in Diagram VIII in Appendix A. The logic would allow for the interrupt for the tracking of the number of times the counter reaches zero, the software-controlled stopping and reading of the timer.

The timer will be initialized and started by software-control. When the timer reaches zero, the execution of the algorithm will be interrupted, a count variable will be incremented, the timer will be reset and restarted, and then the execution of the algorithm

will resume. This will be done without any software-control. When the algorithm is complete, software will stop the timer and read the count value. The software-control will be additional lines of code that will be added to the code for the algorithms. This code will not affect the results of the execution time of the algorithms. After running several tests, I determined the design was sufficient to give effective timing results for the algorithms I would be testing on the SMPPS.

This concluded the hardware design of the system. Now it was time to move on to the development of the algorithms for the system. For this project I will be testing two algorithms. The first will be matrix multiplication and the second would be parallel sorting.

CHAPTER 3

IMPLEMENTATION OF PARALLEL ALGORITHMS

3.1 Matrix Multiplication

3.1.1 Demonstrating a [4x4], [8x8], and [16x16] with [4x4] Matrix

For the matrix-multiplication algorithm (MMA), I wanted to use several different sized matrices to show the effective speed up of using a SMPPS. I would multiply two matrices and place the results in a third matrix. The three matrix sizes I chose were 4x4, 8x8, and a 16x16. This would give me speed up values for simple matrix multiplication that is time-consuming.

I would also produce results for computing the matrix-multiplication on one processor and on the SMPPS. The multiplication of the matrices on the SMPPS would be done in two different ways. One way would be just utilizing the two processors, and the second would utilize the shared-memory. I will be expecting a speed up of almost two for the dual processor system without shared data, and considerably less of a speed up for the shared-memory implementation. This would be caused by the overhead involved in using the SMPPS. The transfer of data through the shared-memory is considerably slower than using registers of a single micro-controller. I do, however, expect a reasonable speed up over the single processor.

I will use 4x4 matrices to demonstrate the different ways I will do the matrix-multiplication algorithm. I will be multiplying matrices A and B, and placing the results in matrix C as shown in **Figure 14**.

Matrix A					Matrix B					Matrix C			
A_{00}	A_{01}	A_{02}	A_{03}	*	B_{00}	B_{01}	B_{02}	B_{03}	=	C_{00}	C_{01}	C_{02}	C_{03}
A_{10}	A_{11}	A_{12}	A_{13}		B_{10}	B_{11}	B_{12}	B_{13}		C_{10}	C_{11}	C_{12}	C_{13}
A_{20}	A_{21}	A_{22}	A_{23}		B_{20}	B_{21}	B_{22}	B_{23}		C_{20}	C_{21}	C_{22}	C_{23}
A_{30}	A_{31}	A_{32}	A_{33}		B_{30}	B_{31}	B_{32}	B_{33}		C_{30}	C_{31}	C_{32}	C_{33}

Figure 14: [4x4] Matrix Multiplication on a single processor.

The operations required to compute Matrix C are shown in **Figure 15**.

$$
\begin{aligned}
C_{00} &= (A_{00}*B_{00})+(A_{01}*B_{10})+(A_{02}*B_{20})+(A_{03}*B_{30})\\
C_{01} &= (A_{00}*B_{01})+(A_{01}*B_{11})+(A_{02}*B_{21})+(A_{03}*B_{31})\\
C_{02} &= (A_{00}*B_{02})+(A_{01}*B_{12})+(A_{02}*B_{22})+(A_{03}*B_{32})\\
C_{03} &= (A_{00}*B_{03})+(A_{01}*B_{13})+(A_{02}*B_{23})+(A_{03}*B_{33})\\
\\
C_{10} &= (A_{10}*B_{00})+(A_{11}*B_{10})+(A_{12}*B_{20})+(A_{13}*B_{30})\\
C_{11} &= (A_{10}*B_{01})+(A_{11}*B_{11})+(A_{12}*B_{21})+(A_{13}*B_{31})\\
C_{12} &= (A_{10}*B_{02})+(A_{11}*B_{12})+(A_{12}*B_{22})+(A_{13}*B_{32})\\
C_{13} &= (A_{10}*B_{03})+(A_{11}*B_{13})+(A_{12}*B_{23})+(A_{13}*B_{33})\\
\\
C_{20} &= (A_{20}*B_{00})+(A_{21}*B_{10})+(A_{22}*B_{20})+(A_{23}*B_{30})\\
C_{21} &= (A_{20}*B_{01})+(A_{21}*B_{11})+(A_{22}*B_{21})+(A_{23}*B_{31})\\
C_{22} &= (A_{20}*B_{02})+(A_{21}*B_{12})+(A_{22}*B_{22})+(A_{23}*B_{32})\\
C_{23} &= (A_{20}*B_{03})+(A_{21}*B_{13})+(A_{22}*B_{23})+(A_{23}*B_{33})\\
\\
C_{30} &= (A_{30}*B_{00})+(A_{31}*B_{10})+(A_{32}*B_{20})+(A_{33}*B_{30})\\
C_{31} &= (A_{30}*B_{01})+(A_{31}*B_{11})+(A_{32}*B_{21})+(A_{33}*B_{31})\\
C_{32} &= (A_{30}*B_{02})+(A_{31}*B_{12})+(A_{32}*B_{22})+(A_{33}*B_{32})\\
C_{33} &= (A_{30}*B_{03})+(A_{31}*B_{13})+(A_{32}*B_{23})+(A_{33}*B_{33})
\end{aligned}
$$

Figure 15: [4x4] Matrix Multiplication.

To obtain the execution time for running the algorithm on one processor, I gave the processor access to all of matrix A and matrix B. The program I developed for this algorithm is in Appendix B. I started out by writing individual programs for each of the three different sized matrices and each of the three different ways. While developing the first few programs, it occurred to me that this might affect the results for the execution times of the algorithm. What I needed was a program that accomplished the three

different types of matrix-multiplication on all three of the matrix sizes. Also, the program must accomplish it with as little different overhead as possible.

As I developed the program, I would test it numerous times. I started to get count values for the different matrices. The values I was getting were very close to the speedups I expected. The problem I was having was that I could not get the program to work exactly like I wanted it to. It would give me results for one matrix size and not the others. As I made changes to correct the problem, another problem would be introduced. Rather then spend tremendous amount of time on trying to resolve these problems. I chose to continue with the writing of the thesis. **Figure 16** shows the results of the execution times.

Matrix Size	One Processor	Dual Processor	Dual Processor Using Shared-Memory
[4x4] Matrix	418	273	386
[8x8] Matrix	1909	1018	1493
[16x16] Matrix	12397	6219	8414

Figure 16: Matrix-Multiplication Execution Times (clock cycles).

The flowchart for the one-processor matrix multiplication algorithm is shown in Diagram IX in Appendix A. In the program the micro-controller would have access to all of matrix A and matrix B. The program would be loaded into the memory of one micro-controller. The program is then started. After the program went through its initializations and loading of variables, the timer would start and it would simply calculate the results for matrix C by the previously stated equations. Once the results were calculated they were moved to shared-memory and the timer was stopped. The last step of the program was to read the values in the timer.

The next step was the program that used two processors to do the matrix multiplication. This was accomplished by giving Processor A access to the first half of matrix A (half of the rows) and access to all of matrix B. Processor A computes the results for the first half of the C matrix. Processor B was given access to the second half of matrix A and all of matrix B. Processor B computes the results for the second half of the C matrix. The dashed line in **Figure 17** shows the separation for the 4x4 matrices:

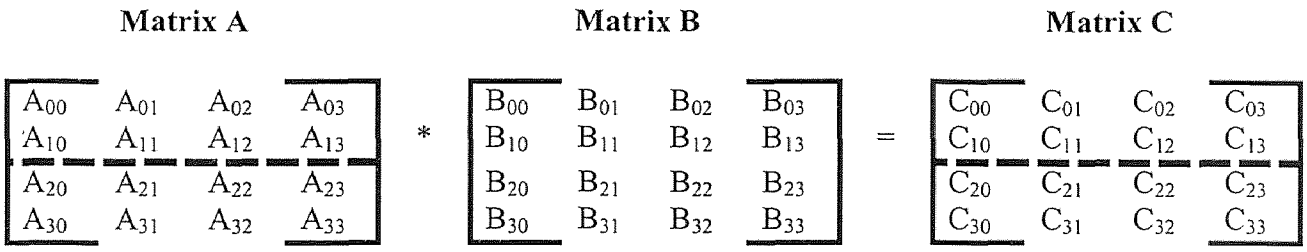


Figure 17: [4x4] Matrix Multiplication on dual processors.

The flowchart for this program is shown in Diagram X in Appendix A. Since the only difference between the program in each processor is what portion of matrix A is accessible, I developed the program to load on the correct portion of the matrix that the individual processor needed. I accomplished this by using a subroutine that required a start and finish location for the values of the matrix. The start and finish locations were determined by which processor was using the program. This was all controlled by the settings placed in the beginning of the program. To gain a better understanding of what I did, a review of the program in Appendix B will be necessary.

In order to obtain the most accurate times as possible, I chose to have the processor control the start and stop of the timer. I accomplished this by using semaphores. These semaphores would be used to signal the other processor when it could continue with its operations. This would allow the initialization and loading of variables by both processors without having to include these operations in the execution times.

Processor A would start by loading its start values and then would enter into a wait state. It would exit that Wait State when Processor B signaled that it had finished loading variables and was now in its own wait state. Now Processor A would start the timer, signal Processor B to start executing, and then start its own execution. Once Processor A completed its execution it would check to see if Processor B was complete. If Processor B were complete, Processor A would stop and read the count value of the timer. Otherwise, Processor A would enter a wait state until Processor B completed its execution.

The final program would give timing results for using shared memory as well as the dual processors. The flowchart for this process is shown in Diagram XI in Appendix A. In this program, both the A matrix and the B matrix are split up. The separation of the matrices is shown in **Figure 18**.

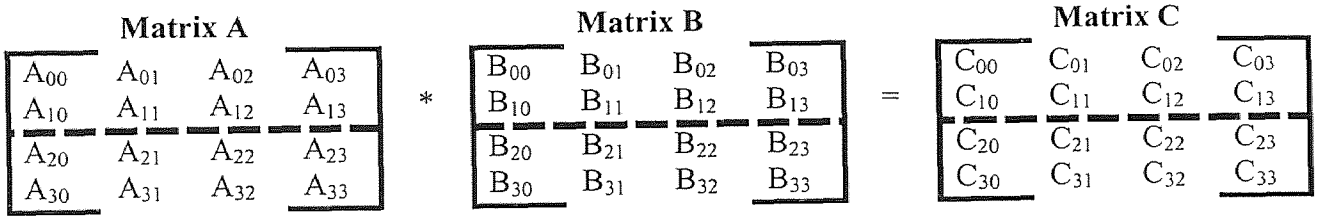


Figure 18: [4x4] Matrix Multiplication on dual processors using shared-memory.

In this program, Processor A has access to the first half of matrix A and the first half of matrix B. Processor A computes the results for the first half of the C matrix. Processor B has access to the second half of matrix A and the second half of matrix B. Processor B computes the results for the second half of the C matrix.

The difference between the program and the dual processor program is that each processor does not have all of the data to complete the computations for the C matrix. For instance, for Processor A to compute the value of C_{00} it would need access to B_{20} and B_{30}. Since Processor B has access to these locations, the data in these locations must be transferred to Processor A through the shared-memory. During the computation portion of the program, each processor must finish the calculations that are possible and wait until it is given the needed data.

I tried to develop the program in a fashion that would allow one processor to make its possible calculations while the other processor was sending and receiving data from the shared-memory. To ensure that a processor did not retrieve the data before it was placed in shared-memory, I used the semaphores to place the processor into a wait state until the required data was available. Once again, a better understanding can be obtained by reviewing the program in Appendix B.

I gave a description on how I implemented the different programs by showing how it was done on a [4x4] matrix. I developed the program to compute the results for the [8x8] matrix. To get the results for the [4x4] case, I added code to reduce the number of loops in the matrix-multiplication routines. I increased the number of loops in the matrix-multiplication routines to get the results for the [16x16]. In order to produce valid timing results, I tried to do this in a way that makes the overall operation of the program

to remain the same for all size matrices. The theory of adding and subtracting loops was sound, but the code to keep the operations the same became quite complex. This is what is causing the delay in the development of a fully operational program.

CHAPTER 4

PERFORMANCE EVALUATIONS

4.1 Matrix Multiplication

As I stated earlier, I am getting consistent results from the current program. However, I am still unable to remove all of the bugs from the program to produce results for all of the program operations. I noticed that overall the results I obtained do not change as I make changes to the program. When I make changes to the program I am able to get results for different size matrices. Several times I was able to get results for more than one size matrix and the results were quite similar to the ones I was getting when I was only able to produce results for one size matrix. Since I am getting results like I expected, I could continue to troubleshoot the current program. With time, I expect to have all the problems worked out of the program. The speedups, based on the results in **Figure 16** are:

Matrix Size	One Processor	Dual Processor	Dual Processor Using Shared-Memory
[4x4] Matrix	418	273	386
[8x8] Matrix	1909	1018	1493
[16x16] Matrix	12307	6219	8414

Figure 16: Matrix-Multiplication Execution Times (clock cycles).

CHAPTER 5

CONCLUSIONS

Based on the results I achieved with the matrix multiplication algorithm, I am concluding that there is an overall effective speedup in using a SMPPS. Overall I would rate this project as a success. I accomplished the first two objectives and made significant progress on the third objective. This project gave me the opportunity to work on a project from the design phase to the testing phase and the opportunity to apply the knowledge I acquired while at NJIT as well as hone my engineering skills.

During the project, I conquered many hurdles and had the chance to have an impact on the curriculum of undergraduate students. Many of the discoveries I made while designing and implementing the micro-controller were beneficial to the EE 393 Lab. Teaching the EE 393 Lab over the summer session was equally rewarding. Not only was I able to increase my understanding of the micro-controller, but I was enabled to impart to the students the knowledge I had gained while working on the project.

The SMPPS project leaves the door open for future areas of study and research. Basing a new system with more processors on this design would present an interesting challenge. Also, developing more parallel algorithms for the system would present en equally challenging obstacle. The possibilities that can be pursued are virtually limitless.

APPENDIX A

DIAGRAMS

Appendix A has the following diagrams:

Dual-Processor Shared-Memory Block Diagram (I)

Dual-Processor Shared-Memory Block Diagram (II)

Original Control Logic Design

1-2 DeMultiplexor Logic

2-1 Multiplexor Logic

Final Shared-Memory Control Logic Design

Default Symbol CTEST Logic

Timer Control Logic

Flow Chart I – One Processor Operation

Flow Chart II – Dual-Processor Operation

Flow Chart III – Dual-Processor Operation using Shared-Memory

Diagram 1

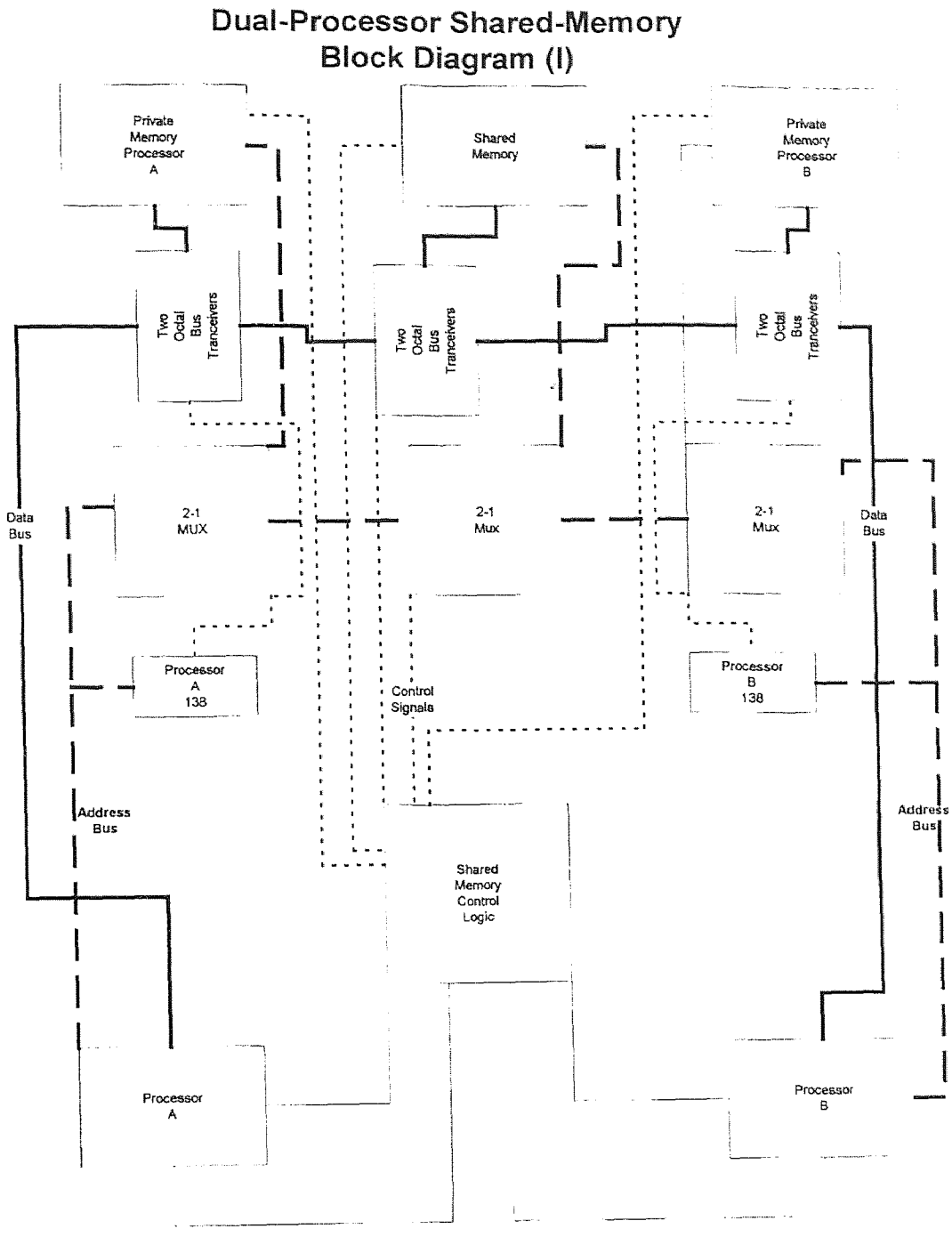

Dual-Processor Shared-Memory
Block Diagram (I)
Private Memory Processor A
Shared Memory
Private Memory Processor B
Two Octal Bus Tranceivers
Two Octal Bus Tranceivers
Two Octal Bus Tranceivers
Data Bus
2-1 MUX
2-1 Mux
2-1 Mux
Data Bus
Processor A 138
Control Signals
Processor B 138
Address Bus
Address Bus
Shared Memory Control Logic
Processor A
Processor B

Diagram 2

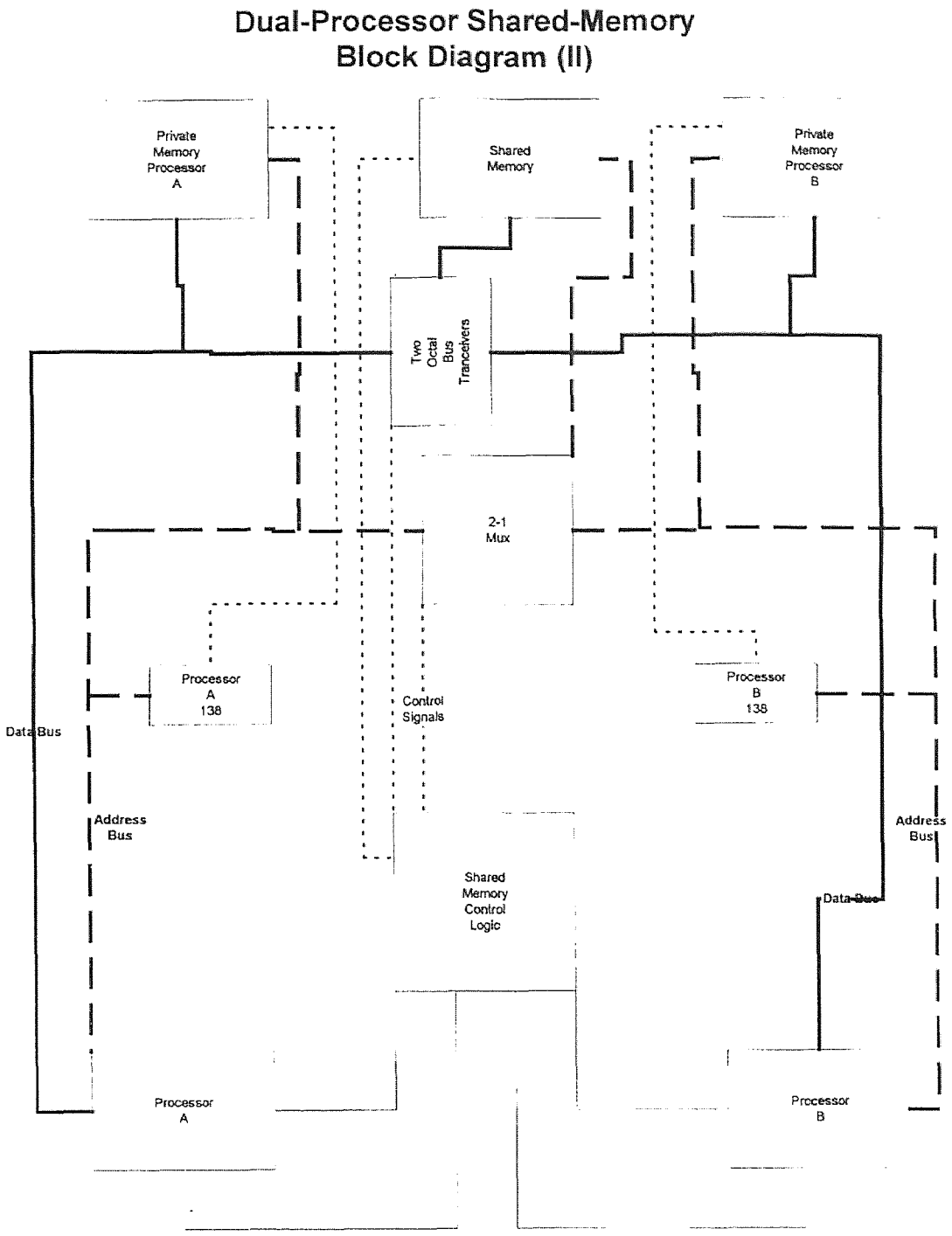

Dual-Processor Shared-Memory
Block Diagram (II)
Private
Memory
Processor
A
Shared
Memory
Private
Memory
Processor
B
Two
Octal
Bus
Tranceivers
2-1
Mux
Processor
A
138
Control
Signals
Processor
B
138
Data Bus
Address
Bus
Address
Bus
Shared
Memory
Control
Logic
Data Bus
Processor
A
Processor
B

Diagram 3

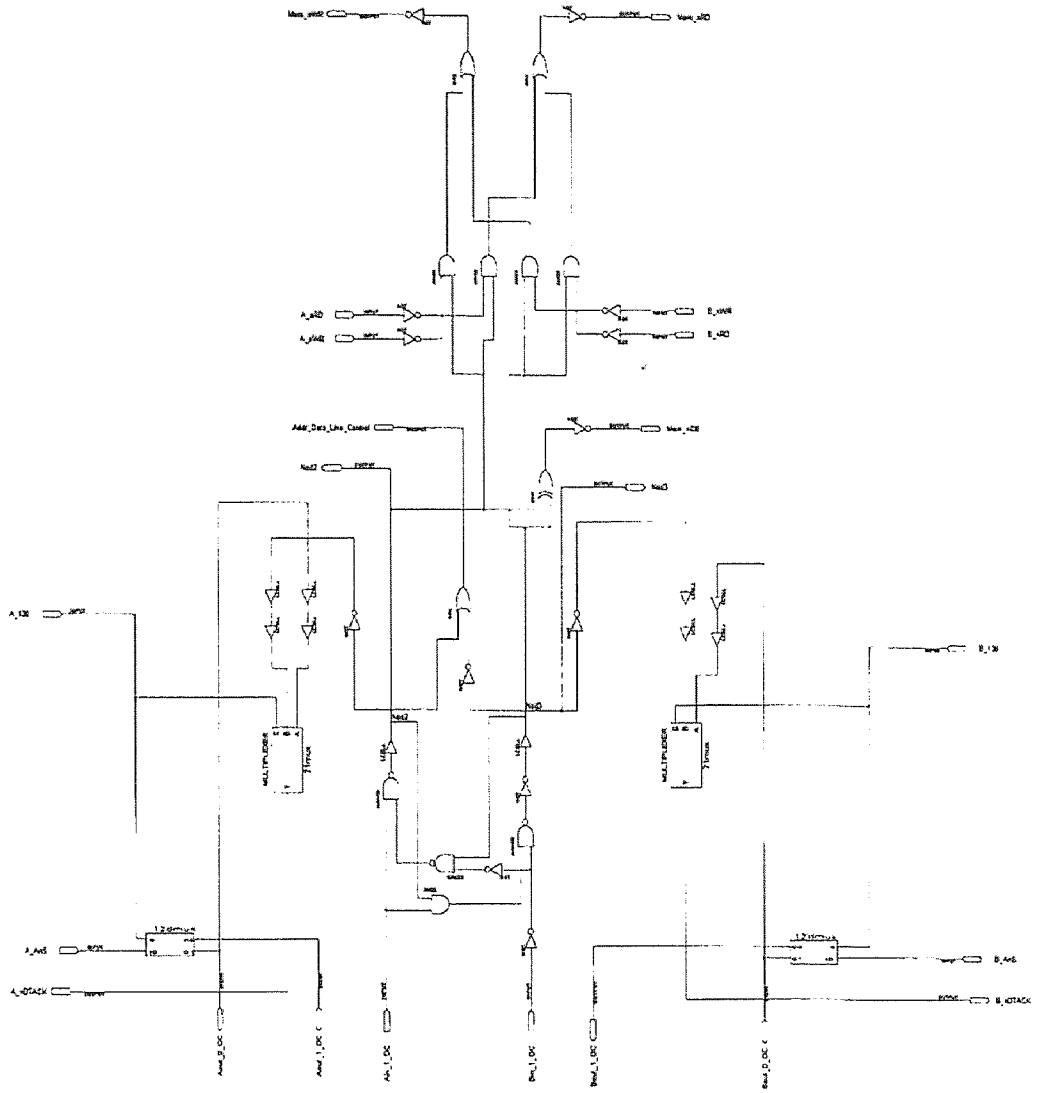
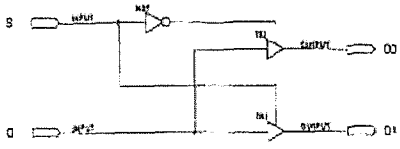


Diagram 4



1-2 DEMUX

TITLE	Shared Memory Logic
COMPANY	NJIT COE Masters
DESIGNER	Eric H. Stoud
SIZE	A
DATE	
TURBO	ON

Diagram 5

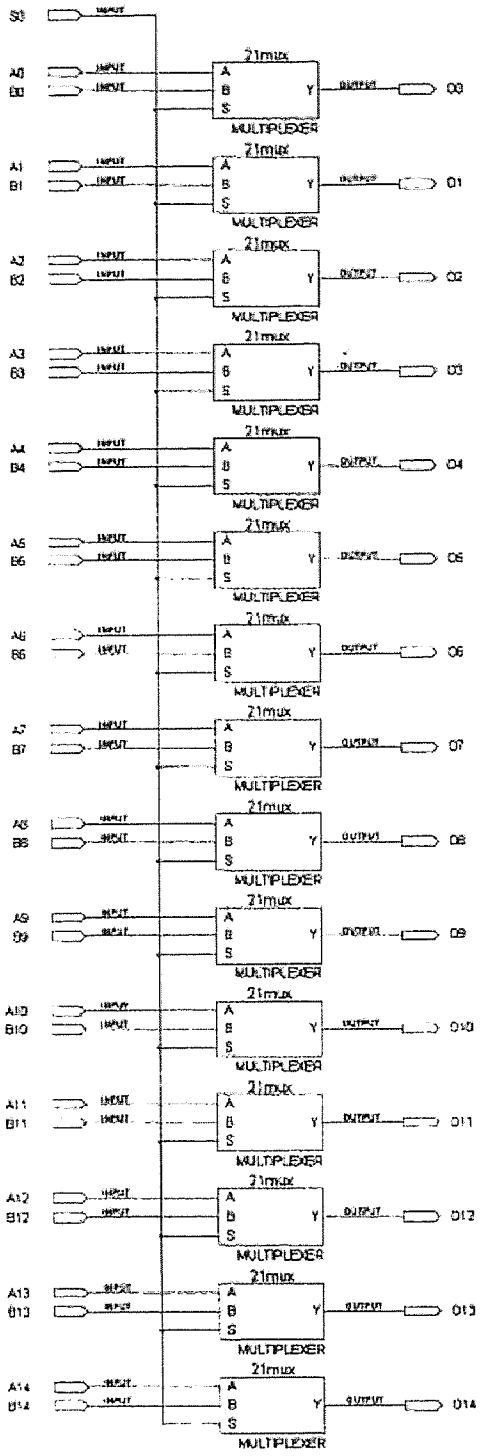
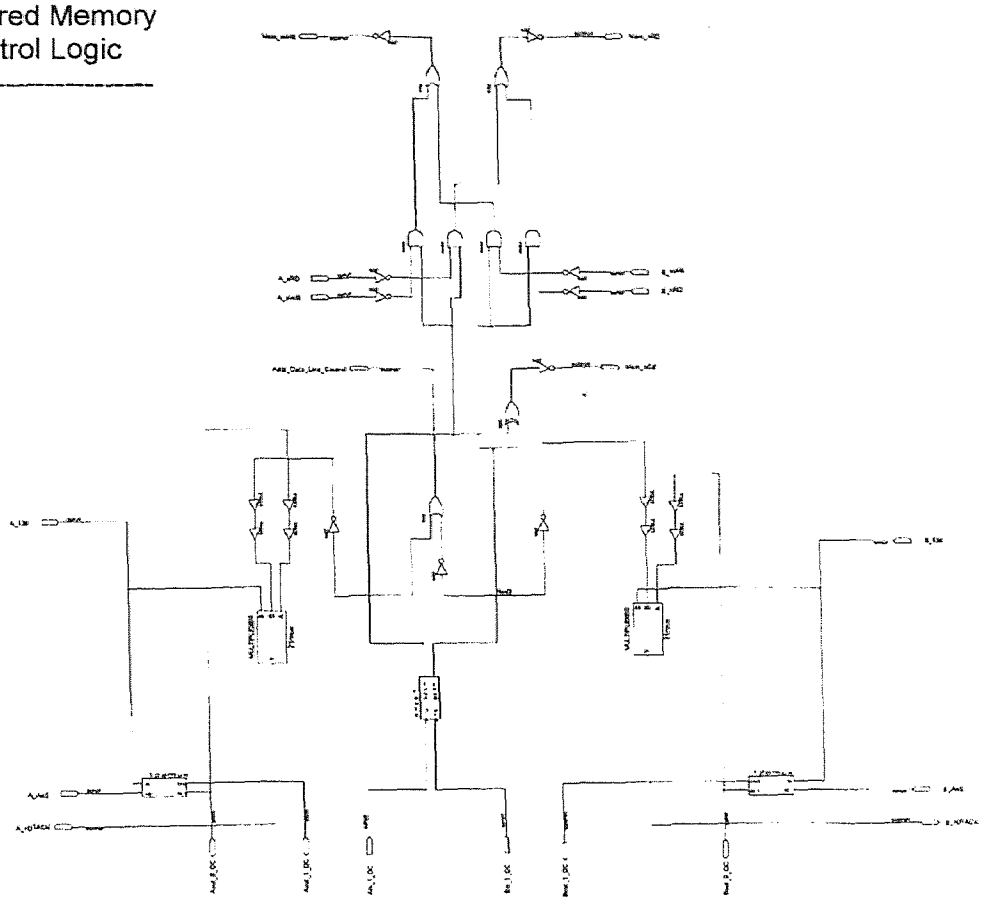

S0
A0
B0
A1
B1
A2
B2
A3
B3
A4
B4
A5
B5
A6
B6
A7
B7
A8
B8
A9
B9
A10
B10
A11
B11
A12
B12
A13
B13
A14
B14
21mux
A
B
S
Y
MULTIPLEXER
O0
O1
O2
O3
O4
O5
O6
O7
O8
O9
O10
O11
O12
O13
O14

Diagram 6

Shared Memory
Control Logic



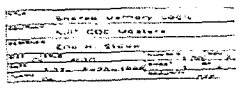
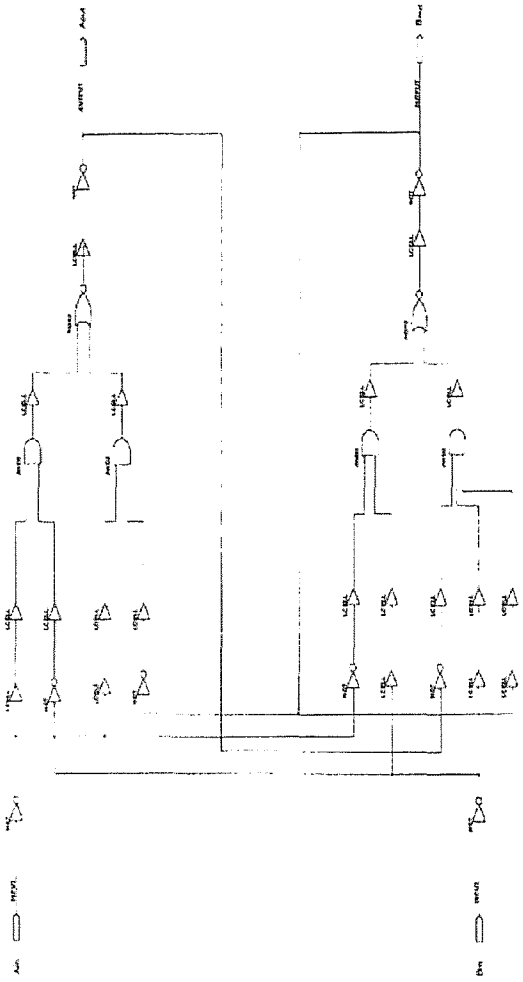


Diagram 7



A/S & /DTACK
Control Logic

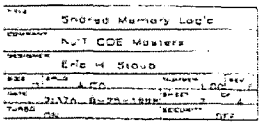


Diagram 8

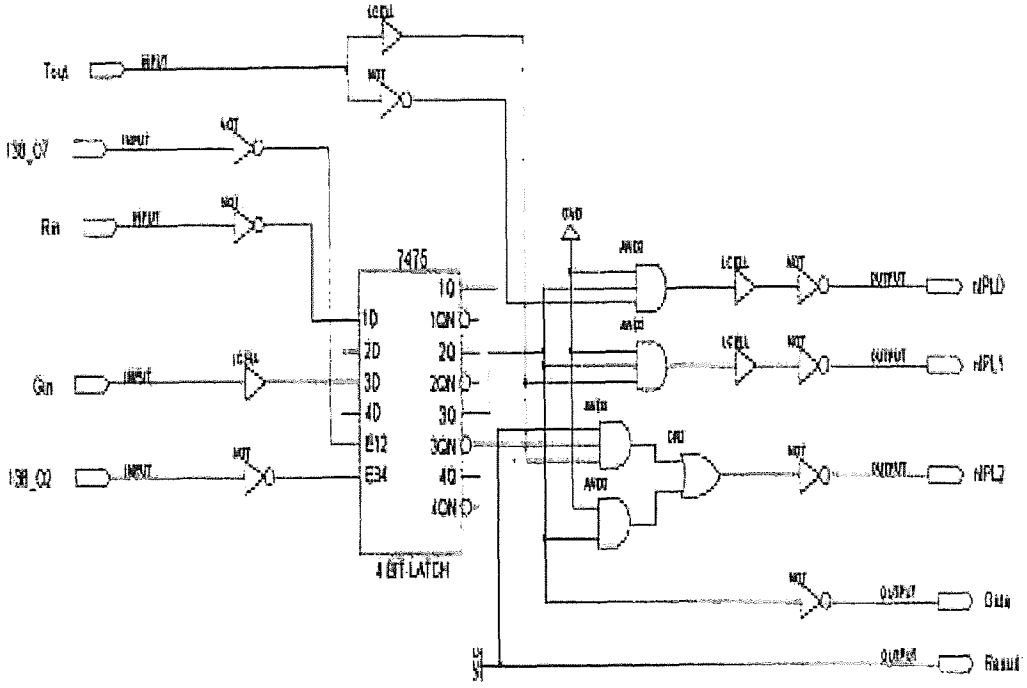

Test
I30_07
Rin
Gin
I38_02
7475
1D
2D
3D
4D
1Q
1QN
2Q
2QN
3Q
3QN
4Q
4QN
4 BIT LATCH
GND
nPL0
nPL1
nPL2
Gout
Rout

Diagram 9

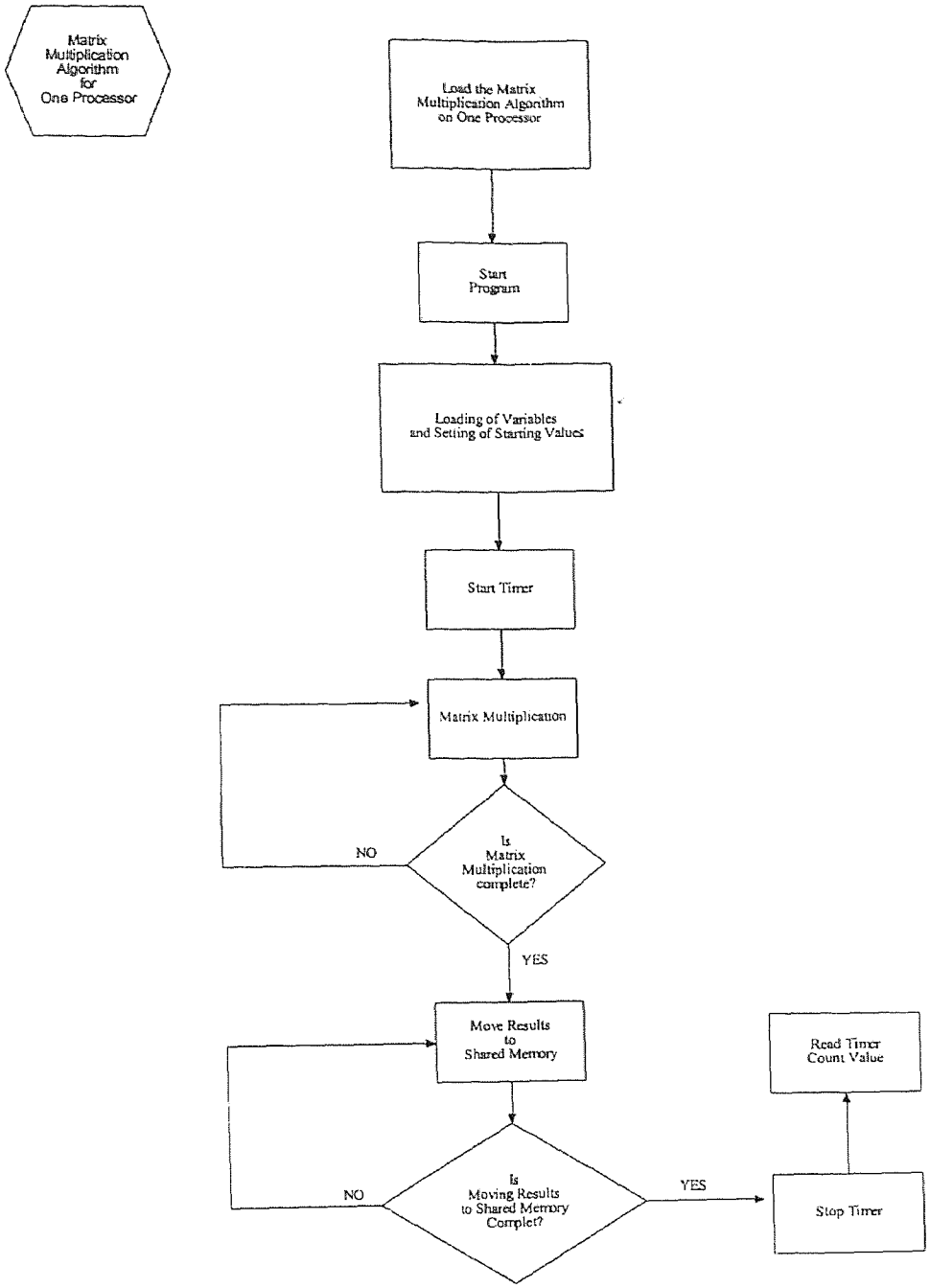

Matrix Multiplication Algorithm for One Processor
Load the Matrix Multiplication Algorithm on One Processor
Start Program
Loading of Variables and Setting of Starting Values
Start Timer
Matrix Multiplication
Is Matrix Multiplication complete?
NO
YES
Move Results to Shared Memory
Is Moving Results to Shared Memory Complet?
NO
YES
Read Timer Count Value
Stop Timer

Diagram 10

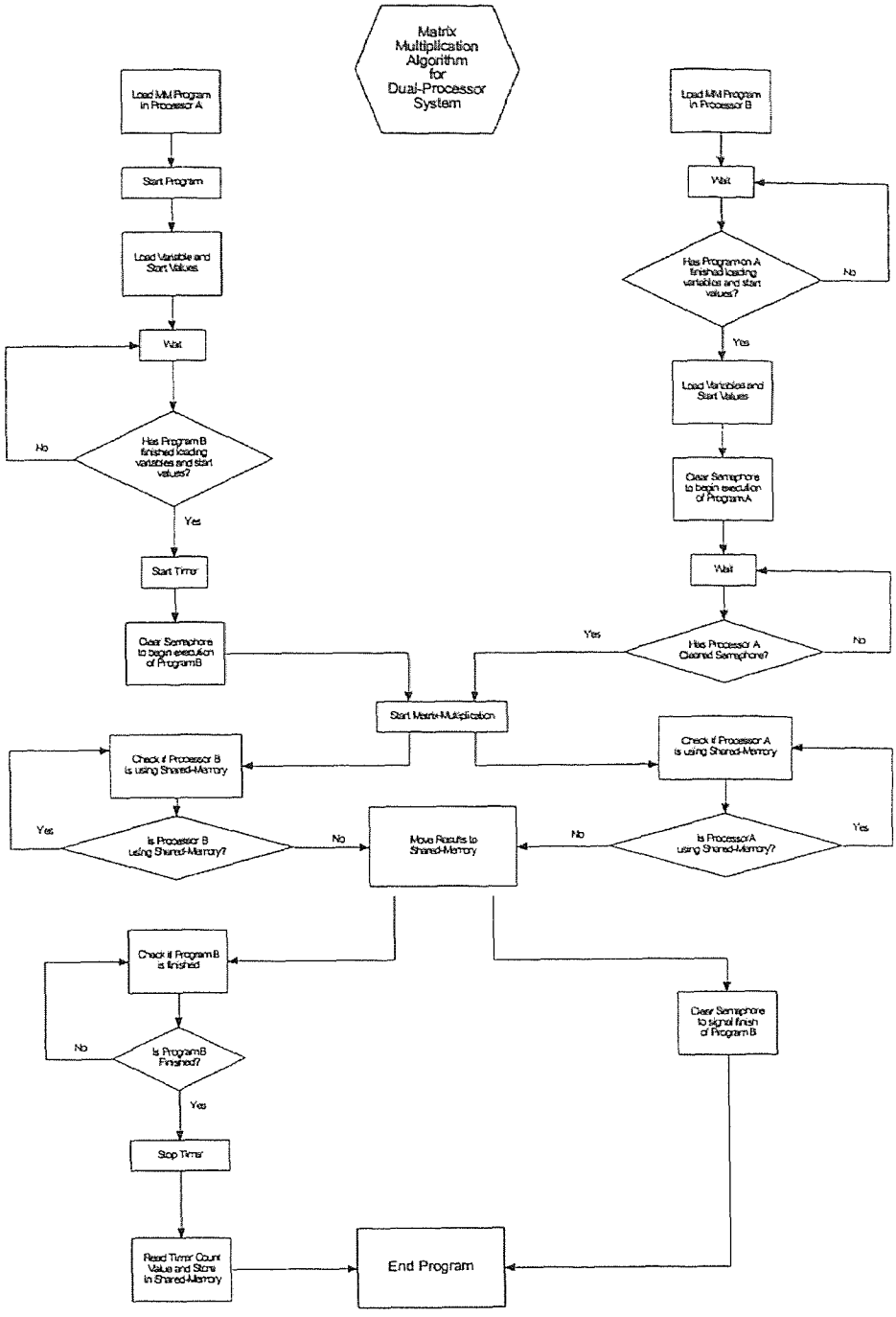
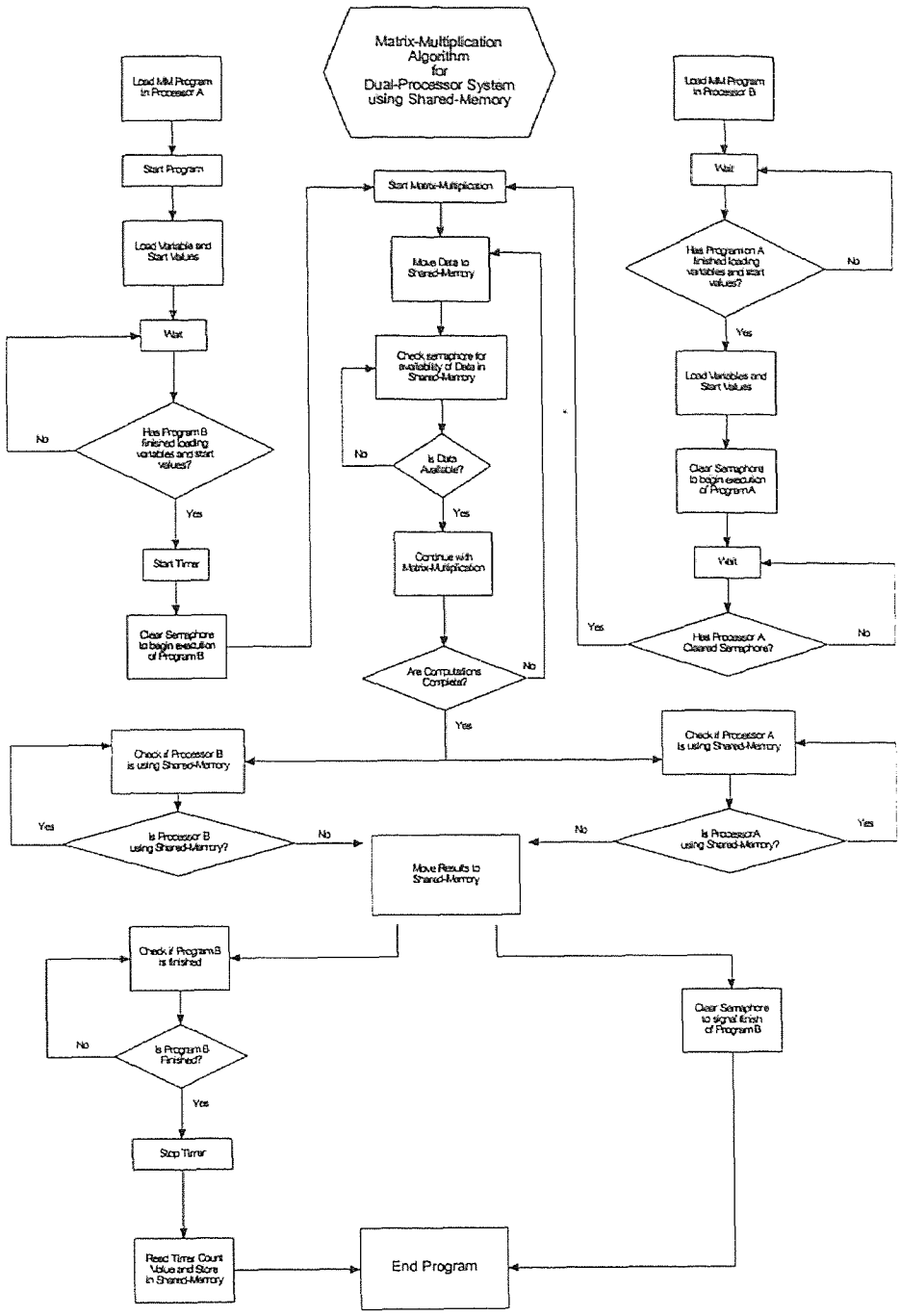

Matrix Multiplication Algorithm for Dual-Processor System
Load MM Program in Processor A
Start Program
Load Variable and Start Values
Wait
Has Program B finished loading variables and start values?
No
Yes
Start Timer
Clear Semaphore to begin execution of Program B
Load MM Program in Processor B
Wait
Has Program A finished loading variables and start values?
No
Yes
Load Variables and Start Values
Clear Semaphore to begin execution of Program A
Wait
Has Processor A Cleared Semaphore?
Yes
No
Start Matrix Multiplication
Check if Processor B is using Shared-Memory
Is Processor B using Shared-Memory?
Yes
No
Check if Processor A is using Shared-Memory
Is Processor A using Shared-Memory?
No
Yes
Move Results to Shared-Memory
Check if Program B is finished
Is Program B Finished?
No
Yes
Clear Semaphore to signal finish of Program B
Stop Timer
Read Timer Count Value and Store in Shared-Memory
End Program

Diagram 11


Matrix-Multiplication Algorithm for Dual-Processor System using Shared-Memory
Load MM Program in Processor A
Start Program
Load Variable and Start Values
Wait
Has Program B finished loading variables and start values?
No
Yes
Start Timer
Clear Semaphore to begin execution of Program B
Start Matrix-Multiplication
Move Data to Shared-Memory
Check semaphore for availability of Data in Shared-Memory
Is Data Available?
No
Yes
Continue with Matrix-Multiplication
Are Computations Complete?
No
Yes
Load MM Program in Processor B
Wait
Has Program on A finished loading variables and start values?
No
Yes
Load Variables and Start Values
Clear Semaphore to begin execution of Program A
Wait
Has Processor A Cleared Semaphore?
No
Yes
Check if Processor B is using Shared-Memory
Is Processor B using Shared-Memory?
Yes
No
Check if Processor A is using Shared-Memory
Is Processor A using Shared-Memory?
No
Yes
Move Results to Shared-Memory
Check if Program B is finished
Is Program B Finished?
No
Yes
Clear Semaphore to signal finish of Program B
Stop Timer
Read Timer Count Value and Store in Shared-Memory
End Program

APPENDIX B

Programs

This is the program for a [4x4], [8x8], and [16x16] Matrix-Multiplication on One Processor System, a Dual-Processor System, and a Dual-Processor System using Shared-Memory.

```
; This is a Matrix Multiplication Algorithm
;
;    PA: Processor A
;    PB: Processor B
;
;    PA[ A11 A12 A13 A14 A15 A16 A17 A18 ]
;    PA[ A21 A22 A23 A24 A15 A16 A17 A18 ]
;    PA[ A31 A32 A33 A34 A15 A16 A17 A18 ]
;    PA[ A41 A42 A43 A44 A15 A16 A17 A18 ]
;    PB[ A51 A52 A53 A54 A15 A16 A17 A18 ]
;    PB[ A61 A62 A63 A64 A15 A16 A17 A18 ]
;    PB[ A71 A72 A73 A74 A15 A16 A17 A18 ]
;    PB[ A81 A82 A83 A84 A15 A16 A17 A18 ]
;
;    PA[ B11 B12 B13 B14 B15 B16 B17 B18 ]
;    PA[ B21 B22 B23 B24 B25 B26 B27 B28 ]
;    PA[ B31 B32 B33 B34 B35 B36 B37 B38 ]
;    PA[ B41 B42 B43 B44 B45 B46 B47 B48 ]
;    PB[ B51 B52 B53 B54 B55 B56 B57 B58 ]
;    PB[ B61 B62 B63 B64 B65 B66 B67 B68 ]
;    PB[ B71 B72 B73 B74 B75 B76 B77 B78 ]
;    PB[ B81 B82 B83 B84 B85 B86 B87 B88 ]
;
; Matrix Type (1)4x4,(2)8x8,(4)16x16
MMT             EQU     $02
; Matrix Shared   (0)NO,1(YES)
MMTS            EQU     $01
; Matrix B Shared (0)NO,(1)YES
MMTSA           EQU     $00
; Program (0)A,(1)B
PROC            EQU     $00
; Matrix Starting Values
ASRT            EQU     $14000
BSRT            EQU     $14100
CSRT            EQU     $14200
LMAVal          EQU     $03
LMBVal          EQU     $03
LMBVla          EQU     $02
; Moving Shared Memory (Start) (End)
MRVA            EQU     $14200
MRVB            EQU     $14240

; Variable Equates
A00             EQU     $14000
A01             EQU     $14001
A02             EQU     $14002
A03             EQU     $14003
A04             EQU     $14004
A05             EQU     $14005
A06             EQU     $14006
A07             EQU     $14007
A08             EQU     $14008
A09             EQU     $14009
A0A             EQU     $1400A
A0B             EQU     $1400B
```

```
A0C             EQU     $1400C
A0D             EQU     $1400D
A0E             EQU     $1400E
A0F             EQU     $1400F
A10             EQU     $14010
A11             EQU     $14011
A12             EQU     $14012
A13             EQU     $14013
A14             EQU     $14014
A15             EQU     $14015
A16             EQU     $14016
A17             EQU     $14017
A18             EQU     $14018
A19             EQU     $14019
A1A             EQU     $1401A
A1B             EQU     $1401B
A1C             EQU     $1401C
A1D             EQU     $1401D
A1E             EQU     $1401E
A1F             EQU     $1401F
A20             EQU     $14020
A21             EQU     $14021
A22             EQU     $14022
A23             EQU     $14023
A24             EQU     $14024
A25             EQU     $14025
A26             EQU     $14026
A27             EQU     $14027
A28             EQU     $14028
A29             EQU     $14029
A2A             EQU     $1402A
A2B             EQU     $1402B
A2C             EQU     $1402C
A2D             EQU     $1402D
A2E             EQU     $1402E
A2F             EQU     $1402F
A30             EQU     $14030
A31             EQU     $14031
A32             EQU     $14032
A33             EQU     $14033
A34             EQU     $14034
A35             EQU     $14035
A36             EQU     $14036
A37             EQU     $14037
A38             EQU     $14038
A39             EQU     $14039
A3A             EQU     $1403A
A3B             EQU     $1403B
A3C             EQU     $1403C
A3D             EQU     $1403D
A3E             EQU     $1403E
A3F             EQU     $1403F
A40             EQU     $14040
A41             EQU     $14041
A42             EQU     $14042
A43             EQU     $14043
A44             EQU     $14044
```

```
A45             EQU     $14045
A46             EQU     $14046
A47             EQU     $14047
A48             EQU     $14048
A49             EQU     $14049
A4A             EQU     $1404A
A4B             EQU     $1404B
A4C             EQU     $1404C
A4D             EQU     $1404D
A4E             EQU     $1404E
A4F             EQU     $1404F
A50             EQU     $14050
A51             EQU     $14051
A52             EQU     $14052
A53             EQU     $14053
A54             EQU     $14054
A55             EQU     $14055
A56             EQU     $14056
A57             EQU     $14057
A58             EQU     $14058
A59             EQU     $14059
A5A             EQU     $1405A
A5B             EQU     $1405B
A5C             EQU     $1405C
A5D             EQU     $1405D
A5E             EQU     $1405E
A5F             EQU     $1405F
A60             EQU     $14060
A61             EQU     $14061
A62             EQU     $14062
A63             EQU     $14063
A64             EQU     $14064
A65             EQU     $14065
A66             EQU     $14066
A67             EQU     $14067
A68             EQU     $14068
A69             EQU     $14069
A6A             EQU     $1406A
A6B             EQU     $1406B
A6C             EQU     $1406C
A6D             EQU     $1406D
A6E             EQU     $1406E
A6F             EQU     $1406F
A70             EQU     $14070
A71             EQU     $14071
A72             EQU     $14072
A73             EQU     $14073
A74             EQU     $14074
A75             EQU     $14075
A76             EQU     $14076
A77             EQU     $14077
A78             EQU     $14078
A79             EQU     $14079
A7A             EQU     $1407A
A7B             EQU     $1407B
A7C             EQU     $1407C
A7D             EQU     $1407D
```

```
A7E             EQU     $1407E
A7F             EQU     $1407F
A80             EQU     $14080
A81             EQU     $14081
A82             EQU     $14082
A83             EQU     $14083
A84             EQU     $14084
A85             EQU     $14085
A86             EQU     $14086
A87             EQU     $14087
A88             EQU     $14088
A89             EQU     $14089
A8A             EQU     $1408A
A8B             EQU     $1408B
A8C             EQU     $1408C
A8D             EQU     $1408D
A8E             EQU     $1408E
A8F             EQU     $1408F
A90             EQU     $14090
A91             EQU     $14091
A92             EQU     $14092
A93             EQU     $14093
A94             EQU     $14094
A95             EQU     $14095
A96             EQU     $14096
A97             EQU     $14097
A98             EQU     $14098
A99             EQU     $14099
A9A             EQU     $1409A
A9B             EQU     $1409B
A9C             EQU     $1409C
A9D             EQU     $1409D
A9E             EQU     $1409E
A9F             EQU     $1409F
AA0             EQU     $140A0
AA1             EQU     $140A1
AA2             EQU     $140A2
AA3             EQU     $140A3
AA4             EQU     $140A4
AA5             EQU     $140A5
AA6             EQU     $140A6
AA7             EQU     $140A7
AA8             EQU     $140A8
AA9             EQU     $140A9
AAA             EQU     $140AA
AAB             EQU     $140AB
AAC             EQU     $140AC
AAD             EQU     $140AD
AAE             EQU     $140AE
AAF             EQU     $140AF
AB0             EQU     $140B0
AB1             EQU     $140B1
AB2             EQU     $140B2
AB3             EQU     $140B3
AB4             EQU     $140B4
AB5             EQU     $140B5
AB6             EQU     $140B6
```

```
AB7             EQU     $140B7
AB8             EQU     $140B8
AB9             EQU     $140B9
ABA             EQU     $140BA
ABB             EQU     $140BB
ABC             EQU     $140BC
ABD             EQU     $140BD
ABE             EQU     $140BE
ABF             EQU     $140BF
AC0             EQU     $140C0
AC1             EQU     $140C1
AC2             EQU     $140C2
AC3             EQU     $140C3
AC4             EQU     $140C4
AC5             EQU     $140C5
AC6             EQU     $140C6
AC7             EQU     $140C7
AC8             EQU     $140C8
AC9             EQU     $140C9
ACA             EQU     $140CA
ACB             EQU     $140CB
ACC             EQU     $140CC
ACD             EQU     $140CD
ACE             EQU     $140CE
ACF             EQU     $140CF
AD0             EQU     $140D0
AD1             EQU     $140D1
AD2             EQU     $140D2
AD3             EQU     $140D3
AD4             EQU     $140D4
AD5             EQU     $140D5
AD6             EQU     $140D6
AD7             EQU     $140D7
AD8             EQU     $140D8
AD9             EQU     $140D9
ADA             EQU     $140DA
ADB             EQU     $140DB
ADC             EQU     $140DC
ADD             EQU     $140DD
ADE             EQU     $140DE
ADF             EQU     $140DF
AE0             EQU     $140E0
AE1             EQU     $140E1
AE2             EQU     $140E2
AE3             EQU     $140E3
AE4             EQU     $140E4
AE5             EQU     $140E5
AE6             EQU     $140E6
AE7             EQU     $140E7
AE8             EQU     $140E8
AE9             EQU     $140E9
AEA             EQU     $140EA
AEB             EQU     $140EB
AEC             EQU     $140EC
AED             EQU     $140ED
AEE             EQU     $140EE
AEF             EQU     $140EF
```

```
AF0             EQU     $140F0
AF1             EQU     $140F1
AF2             EQU     $140F2
AF3             EQU     $140F3
AF4             EQU     $140F4
AF5             EQU     $140F5
AF6             EQU     $140F6
AF7             EQU     $140F7
AF8             EQU     $140F8
AF9             EQU     $140F9
AFA             EQU     $140FA
AFB             EQU     $140FB
AFC             EQU     $140FC
AFD             EQU     $140FD
AFE             EQU     $140FE
AFF             EQU     $140FF
;
B00             EQU     $14100
B10             EQU     $14101
B20             EQU     $14102
B30             EQU     $14103
B40             EQU     $14104
B50             EQU     $14105
B60             EQU     $14106
B70             EQU     $14107
B80             EQU     $14108
B90             EQU     $14109
BA0             EQU     $1410A
BB0             EQU     $1410B
BC0             EQU     $1410C
BD0             EQU     $1410D
BE0             EQU     $1410E
BF0             EQU     $1410F
B01             EQU     $14110
B11             EQU     $14111
B21             EQU     $14112
B31             EQU     $14113
B41             EQU     $14114
B51             EQU     $14115
B61             EQU     $14116
B71             EQU     $14117
B81             EQU     $14118
B91             EQU     $14119
BA1             EQU     $1411A
BB1             EQU     $1411B
BC1             EQU     $1411C
BD1             EQU     $1411D
BE1             EQU     $1411E
BF1             EQU     $1411F
B02             EQU     $14120
B12             EQU     $14121
B22             EQU     $14122
B32             EQU     $14123
B42             EQU     $14124
B52             EQU     $14125
B62             EQU     $14126
B72             EQU     $14127
```

```
B82             EQU     $14128
B92             EQU     $14129
BA2             EQU     $1412A
BB2             EQU     $1412B
BC2             EQU     $1412C
BD2             EQU     $1412D
BE2             EQU     $1412E
BF2             EQU     $1412F
B03             EQU     $14130
B13             EQU     $14131
B23             EQU     $14132
B33             EQU     $14133
B43             EQU     $14134
B53             EQU     $14135
B63             EQU     $14136
B73             EQU     $14137
B83             EQU     $14138
B93             EQU     $14139
BA3             EQU     $1413A
BB3             EQU     $1413B
BC3             EQU     $1413C
BD3             EQU     $1413D
BE3             EQU     $1413E
BF3             EQU     $1413F
B04             EQU     $14140
B14             EQU     $14141
B24             EQU     $14142
B34             EQU     $14143
B44             EQU     $14144
B54             EQU     $14145
B64             EQU     $14146
B74             EQU     $14147
B84             EQU     $14148
B94             EQU     $14149
BA4             EQU     $1414A
BB4             EQU     $1414B
BC4             EQU     $1414C
BD4             EQU     $1414D
BE4             EQU     $1414E
BF4             EQU     $1414F
B05             EQU     $14150
B15             EQU     $14151
B25             EQU     $14152
B35             EQU     $14153
B45             EQU     $14154
B55             EQU     $14155
B65             EQU     $14156
B75             EQU     $14157
B85             EQU     $14158
B95             EQU     $14159
BA5             EQU     $1415A
BB5             EQU     $1415B
BC5             EQU     $1415C
BD5             EQU     $1415D
BE5             EQU     $1415E
BF5             EQU     $1415F
B06             EQU     $14160
```

```
B16             EQU     $14161
B26             EQU     $14162
B36             EQU     $14163
B46             EQU     $14164
B56             EQU     $14165
B66             EQU     $14166
B76             EQU     $14167
B86             EQU     $14168
B96             EQU     $14169
BA6             EQU     $1416A
BB6             EQU     $1416B
BC6             EQU     $1416C
BD6             EQU     $1416D
BE6             EQU     $1416E
BF6             EQU     $1416F
B07             EQU     $14170
B17             EQU     $14171
B27             EQU     $14172
B37             EQU     $14173
B47             EQU     $14174
B57             EQU     $14175
B67             EQU     $14176
B77             EQU     $14177
B87             EQU     $14178
B97             EQU     $14179
BA7             EQU     $1417A
BB7             EQU     $1417B
BC7             EQU     $1417C
BD7             EQU     $1417D
BE7             EQU     $1417E
BF7             EQU     $1417F
B08             EQU     $14180
B18             EQU     $14181
B28             EQU     $14182
B38             EQU     $14183
B48             EQU     $14184
B58             EQU     $14185
B68             EQU     $14186
B78             EQU     $14187
B88             EQU     $14188
B98             EQU     $14189
BA8             EQU     $1418A
BB8             EQU     $1418B
BC8             EQU     $1418C
BD8             EQU     $1418D
BE8             EQU     $1418E
BF8             EQU     $1418F
B09             EQU     $14190
B19             EQU     $14191
B29             EQU     $14192
B39             EQU     $14193
B49             EQU     $14194
B59             EQU     $14195
B69             EQU     $14196
B79             EQU     $14197
B89             EQU     $14198
B99             EQU     $14199
```

```
BA9             EQU     $1419A
BB9             EQU     $1419B
BC9             EQU     $1419C
BD9             EQU     $1419D
BE9             EQU     $1419E
BF9             EQU     $1419F
B0A             EQU     $141A0
B1A             EQU     $141A1
B2A             EQU     $141A2
B3A             EQU     $141A3
B4A             EQU     $141A4
B5A             EQU     $141A5
B6A             EQU     $141A6
B7A             EQU     $141A7
B8A             EQU     $141A8
B9A             EQU     $141A9
BAA             EQU     $141AA
BBA             EQU     $141AB
BCA             EQU     $141AC
BDA             EQU     $141AD
BEA             EQU     $141AE
BFA             EQU     $141AF
B0B             EQU     $141B0
B1B             EQU     $141B1
B2B             EQU     $141B2
B3B             EQU     $141B3
B4B             EQU     $141B4
B5B             EQU     $141B5
B6B             EQU     $141B6
B7B             EQU     $141B7
B8B             EQU     $141B8
B9B             EQU     $141B9
BAB             EQU     $141BA
BBB             EQU     $141BB
BCB             EQU     $141BC
BDB             EQU     $141BD
BEB             EQU     $141BE
BFB             EQU     $141BF
B0C             EQU     $141C0
B1C             EQU     $141C1
B2C             EQU     $141C2
B3C             EQU     $141C3
B4C             EQU     $141C4
B5C             EQU     $141C5
B6C             EQU     $141C6
B7C             EQU     $141C7
B8C             EQU     $141C8
B9C             EQU     $141C9
BAC             EQU     $141CA
BBC             EQU     $141CB
BCC             EQU     $141CC
BDC             EQU     $141CD
BEC             EQU     $141CE
BFC             EQU     $141CF
B0D             EQU     $141D0
B1D             EQU     $141D1
B2D             EQU     $141D2
```

```
B3D             EQU     $141D3
B4D             EQU     $141D4
B5D             EQU     $141D5
B6D             EQU     $141D6
B7D             EQU     $141D7
B8D             EQU     $141D8
B9D             EQU     $141D9
BAD             EQU     $141DA
BBD             EQU     $141DB
BCD             EQU     $141DC
BDD             EQU     $141DD
BED             EQU     $141DE
BFD             EQU     $141DF
B0E             EQU     $141E0
B1E             EQU     $141E1
B2E             EQU     $141E2
B3E             EQU     $141E3
B4E             EQU     $141E4
B5E             EQU     $141E5
B6E             EQU     $141E6
B7E             EQU     $141E7
B8E             EQU     $141E8
B9E             EQU     $141E9
BAE             EQU     $141EA
BBE             EQU     $141EB
BCE             EQU     $141EC
BDE             EQU     $141ED
BEE             EQU     $141EE
BFE             EQU     $141EF
B0F             EQU     $141F0
B1F             EQU     $141F1
B2F             EQU     $141F2
B3F             EQU     $141F3
B4F             EQU     $141F4
B5F             EQU     $141F5
B6F             EQU     $141F6
B7F             EQU     $141F7
B8F             EQU     $141F8
B9F             EQU     $141F9
BAF             EQU     $141FA
BBF             EQU     $141FB
BCF             EQU     $141FC
BDF             EQU     $141FD
BEF             EQU     $141FE
BFF             EQU     $141FF
;
C00             EQU     $14200
C01             EQU     $14201
C02             EQU     $14202
C03             EQU     $14203
C04             EQU     $14204
C05             EQU     $14205
C06             EQU     $14206
C07             EQU     $14207
C08             EQU     $14208
C09             EQU     $14209
C0A             EQU     $1420A
```

```
C0B             EQU     $1420B
C0C             EQU     $1420C
C0D             EQU     $1420D
C0E             EQU     $1420E
C0F             EQU     $1420F
C10             EQU     $14210
C11             EQU     $14211
C12             EQU     $14212
C13             EQU     $14213
C14             EQU     $14214
C15             EQU     $14215
C16             EQU     $14216
C17             EQU     $14217
C18             EQU     $14218
C19             EQU     $14219
C1A             EQU     $1421A
C1B             EQU     $1421B
C1C             EQU     $1421C
C1D             EQU     $1421D
C1E             EQU     $1421E
C1F             EQU     $1421F
C20             EQU     $14220
C21             EQU     $14221
C22             EQU     $14222
C23             EQU     $14223
C24             EQU     $14224
C25             EQU     $14225
C26             EQU     $14226
C27             EQU     $14227
C28             EQU     $14228
C29             EQU     $14229
C2A             EQU     $1422A
C2B             EQU     $1422B
C2C             EQU     $1422C
C2D             EQU     $1422D
C2E             EQU     $1422E
C2F             EQU     $1422F
C30             EQU     $14230
C31             EQU     $14231
C32             EQU     $14232
C33             EQU     $14233
C34             EQU     $14234
C35             EQU     $14235
C36             EQU     $14236
C37             EQU     $14237
C38             EQU     $14238
C39             EQU     $14239
C3A             EQU     $1423A
C3B             EQU     $1423B
C3C             EQU     $1423C
C3D             EQU     $1423D
C3E             EQU     $1423E
C3F             EQU     $1423F
C40             EQU     $14240
C41             EQU     $14241
C42             EQU     $14242
C43             EQU     $14243
```

```
C44             EQU     $14244
C45             EQU     $14245
C46             EQU     $14246
C47             EQU     $14247
C48             EQU     $14248
C49             EQU     $14249
C4A             EQU     $1424A
C4B             EQU     $1424B
C4C             EQU     $1424C
C4D             EQU     $1424D
C4E             EQU     $1424E
C4F             EQU     $1424F
C50             EQU     $14250
C51             EQU     $14251
C52             EQU     $14252
C53             EQU     $14253
C54             EQU     $14254
C55             EQU     $14255
C56             EQU     $14256
C57             EQU     $14257
C58             EQU     $14258
C59             EQU     $14259
C5A             EQU     $1425A
C5B             EQU     $1425B
C5C             EQU     $1425C
C5D             EQU     $1425D
C5E             EQU     $1425E
C5F             EQU     $1425F
C60             EQU     $14260
C61             EQU     $14261
C62             EQU     $14262
C63             EQU     $14263
C64             EQU     $14264
C65             EQU     $14265
C66             EQU     $14266
C67             EQU     $14267
C68             EQU     $14268
C69             EQU     $14269
C6A             EQU     $1426A
C6B             EQU     $1426B
C6C             EQU     $1426C
C6D             EQU     $1426D
C6E             EQU     $1426E
C6F             EQU     $1426F
C70             EQU     $14270
C71             EQU     $14271
C72             EQU     $14272
C73             EQU     $14273
C74             EQU     $14274
C75             EQU     $14275
C76             EQU     $14276
C77             EQU     $14277
C78             EQU     $14278
C79             EQU     $14279
C7A             EQU     $1427A
C7B             EQU     $1427B
C7C             EQU     $1427C
```

```
C7D             EQU     $1427D
C7E             EQU     $1427E
C7F             EQU     $1427F
C80             EQU     $14280
C81             EQU     $14281
C82             EQU     $14282
C83             EQU     $14283
C84             EQU     $14284
C85             EQU     $14285
C86             EQU     $14286
C87             EQU     $14287
C88             EQU     $14288
C89             EQU     $14289
C8A             EQU     $1428A
C8B             EQU     $1428B
C8C             EQU     $1428C
C8D             EQU     $1428D
C8E             EQU     $1428E
C8F             EQU     $1428F
C90             EQU     $14290
C91             EQU     $14291
C92             EQU     $14292
C93             EQU     $14293
C94             EQU     $14294
C95             EQU     $14295
C96             EQU     $14296
C97             EQU     $14297
C98             EQU     $14298
C99             EQU     $14299
C9A             EQU     $1429A
C9B             EQU     $1429B
C9C             EQU     $1429C
C9D             EQU     $1429D
C9E             EQU     $1429E
C9F             EQU     $1429F
CA0             EQU     $142A0
CA1             EQU     $142A1
CA2             EQU     $142A2
CA3             EQU     $142A3
CA4             EQU     $142A4
CA5             EQU     $142A5
CA6             EQU     $142A6
CA7             EQU     $142A7
CA8             EQU     $142A8
CA9             EQU     $142A9
CAA             EQU     $142AA
CAB             EQU     $142AB
CAC             EQU     $142AC
CAD             EQU     $142AD
CAE             EQU     $142AE
CAF             EQU     $142AF
CB0             EQU     $142B0
CB1             EQU     $142B1
CB2             EQU     $142B2
CB3             EQU     $142B3
CB4             EQU     $142B4
CB5             EQU     $142B5
```

```
CB6             EQU     $142B6
CB7             EQU     $142B7
CB8             EQU     $142B8
CB9             EQU     $142B9
CBA             EQU     $142BA
CBB             EQU     $142BB
CBC             EQU     $142BC
CBD             EQU     $142BD
CBE             EQU     $142BE
CBF             EQU     $142BF
CC0             EQU     $142C0
CC1             EQU     $142C1
CC2             EQU     $142C2
CC3             EQU     $142C3
CC4             EQU     $142C4
CC5             EQU     $142C5
CC6             EQU     $142C6
CC7             EQU     $142C7
CC8             EQU     $142C8
CC9             EQU     $142C9
CCA             EQU     $142CA
CCB             EQU     $142CB
CCC             EQU     $142CC
CCD             EQU     $142CD
CCE             EQU     $142CE
CCF             EQU     $142CF
CD0             EQU     $142D0
CD1             EQU     $142D1
CD2             EQU     $142D2
CD3             EQU     $142D3
CD4             EQU     $142D4
CD5             EQU     $142D5
CD6             EQU     $142D6
CD7             EQU     $142D7
CD8             EQU     $142D8
CD9             EQU     $142D9
CDA             EQU     $142DA
CDB             EQU     $142DB
CDC             EQU     $142DC
CDD             EQU     $142DD
CDE             EQU     $142DE
CDF             EQU     $142DF
CE0             EQU     $142E0
CE1             EQU     $142E1
CE2             EQU     $142E2
CE3             EQU     $142E3
CE4             EQU     $142E4
CE5             EQU     $142E5
CE6             EQU     $142E6
CE7             EQU     $142E7
CE8             EQU     $142E8
CE9             EQU     $142E9
CEA             EQU     $142EA
CEB             EQU     $142EB
CEC             EQU     $142EC
CED             EQU     $142ED
CEE             EQU     $142EE
```

```
CEF             EQU     $142EF
CF0             EQU     $142F0
CF1             EQU     $142F1
CF2             EQU     $142F2
CF3             EQU     $142F3
CF4             EQU     $142F4
CF5             EQU     $142F5
CF6             EQU     $142F6
CF7             EQU     $142F7
CF8             EQU     $142F8
CF9             EQU     $142F9
CFA             EQU     $142FA
CFB             EQU     $142FB
CFC             EQU     $142FC
CFD             EQU     $142FD
CFE             EQU     $142FE
CFF             EQU     $142FF
;
SA00             EQU     $28000
SA01             EQU     $28001
SA02             EQU     $28002
SA03             EQU     $28003
SA04             EQU     $28004
SA05             EQU     $28005
SA06             EQU     $28006
SA07             EQU     $28007
SA08             EQU     $28008
SA09             EQU     $28009
SA0A             EQU     $2800A
SA0B             EQU     $2800B
SA0C             EQU     $2800C
SA0D             EQU     $2800D
SA0E             EQU     $2800E
SA0F             EQU     $2800F
SA10             EQU     $28010
SA11             EQU     $28011
SA12             EQU     $28012
SA13             EQU     $28013
SA14             EQU     $28014
SA15             EQU     $28015
SA16             EQU     $28016
SA17             EQU     $28017
SA18             EQU     $28018
SA19             EQU     $28019
SA1A             EQU     $2801A
SA1B             EQU     $2801B
SA1C             EQU     $2801C
SA1D             EQU     $2801D
SA1E             EQU     $2801E
SA1F             EQU     $2801F
SA20             EQU     $28020
SA21             EQU     $28021
SA22             EQU     $28022
SA23             EQU     $28023
SA24             EQU     $28024
SA25             EQU     $28025
SA26             EQU     $28026
```

```
SA27            EQU     $28027
SA28            EQU     $28028
SA29            EQU     $28029
SA2A            EQU     $2802A
SA2B            EQU     $2802B
SA2C            EQU     $2802C
SA2D            EQU     $2802D
SA2E            EQU     $2802E
SA2F            EQU     $2802F
SA30            EQU     $28030
SA31            EQU     $28031
SA32            EQU     $28032
SA33            EQU     $28033
SA34            EQU     $28034
SA35            EQU     $28035
SA36            EQU     $28036
SA37            EQU     $28037
SA38            EQU     $28038
SA39            EQU     $28039
SA3A            EQU     $2803A
SA3B            EQU     $2803B
SA3C            EQU     $2803C
SA3D            EQU     $2803D
SA3E            EQU     $2803E
SA3F            EQU     $2803F
SA40            EQU     $28040
SA41            EQU     $28041
SA42            EQU     $28042
SA43            EQU     $28043
SA44            EQU     $28044
SA45            EQU     $28045
SA46            EQU     $28046
SA47            EQU     $28047
SA48            EQU     $28048
SA49            EQU     $28049
SA4A            EQU     $2804A
SA4B            EQU     $2804B
SA4C            EQU     $2804C
SA4D            EQU     $2804D
SA4E            EQU     $2804E
SA4F            EQU     $2804F
SA50            EQU     $28050
SA51            EQU     $28051
SA52            EQU     $28052
SA53            EQU     $28053
SA54            EQU     $28054
SA55            EQU     $28055
SA56            EQU     $28056
SA57            EQU     $28057
SA58            EQU     $28058
SA59            EQU     $28059
SA5A            EQU     $2805A
SA5B            EQU     $2805B
SA5C            EQU     $2805C
SA5D            EQU     $2805D
SA5E            EQU     $2805E
SA5F            EQU     $2805F
```

```
SA60            EQU     $28060
SA61            EQU     $28061
SA62            EQU     $28062
SA63            EQU     $28063
SA64            EQU     $28064
SA65            EQU     $28065
SA66            EQU     $28066
SA67            EQU     $28067
SA68            EQU     $28068
SA69            EQU     $28069
SA6A            EQU     $2806A
SA6B            EQU     $2806B
SA6C            EQU     $2806C
SA6D            EQU     $2806D
SA6E            EQU     $2806E
SA6F            EQU     $2806F
SA70            EQU     $28070
SA71            EQU     $28071
SA72            EQU     $28072
SA73            EQU     $28073
SA74            EQU     $28074
SA75            EQU     $28075
SA76            EQU     $28076
SA77            EQU     $28077
SA78            EQU     $28078
SA79            EQU     $28079
SA7A            EQU     $2807A
SA7B            EQU     $2807B
SA7C            EQU     $2807C
SA7D            EQU     $2807D
SA7E            EQU     $2807E
SA7F            EQU     $2807F
SA80            EQU     $28080
SA81            EQU     $28081
SA82            EQU     $28082
SA83            EQU     $28083
SA84            EQU     $28084
SA85            EQU     $28085
SA86            EQU     $28086
SA87            EQU     $28087
SA88            EQU     $28088
SA89            EQU     $28089
SA8A            EQU     $2808A
SA8B            EQU     $2808B
SA8C            EQU     $2808C
SA8D            EQU     $2808D
SA8E            EQU     $2808E
SA8F            EQU     $2808F
SA90            EQU     $28090
SA91            EQU     $28091
SA92            EQU     $28092
SA93            EQU     $28093
SA94            EQU     $28094
SA95            EQU     $28095
SA96            EQU     $28096
SA97            EQU     $28097
SA98            EQU     $28098
```

```
SA99            EQU     $28099
SA9A            EQU     $2809A
SA9B            EQU     $2809B
SA9C            EQU     $2809C
SA9D            EQU     $2809D
SA9E            EQU     $2809E
SA9F            EQU     $2809F
SAA0            EQU     $280A0
SAA1            EQU     $280A1
SAA2            EQU     $280A2
SAA3            EQU     $280A3
SAA4            EQU     $280A4
SAA5            EQU     $280A5
SAA6            EQU     $280A6
SAA7            EQU     $280A7
SAA8            EQU     $280A8
SAA9            EQU     $280A9
SAAA            EQU     $280AA
SAAB            EQU     $280AB
SAAC            EQU     $280AC
SAAD            EQU     $280AD
SAAE            EQU     $280AE
SAAF            EQU     $280AF
SAB0            EQU     $280B0
SAB1            EQU     $280B1
SAB2            EQU     $280B2
SAB3            EQU     $280B3
SAB4            EQU     $280B4
SAB5            EQU     $280B5
SAB6            EQU     $280B6
SAB7            EQU     $280B7
SAB8            EQU     $280B8
SAB9            EQU     $280B9
SABA            EQU     $280BA
SABB            EQU     $280BB
SABC            EQU     $280BC
SABD            EQU     $280BD
SABE            EQU     $280BE
SABF            EQU     $280BF
SAC0            EQU     $280C0
SAC1            EQU     $280C1
SAC2            EQU     $280C2
SAC3            EQU     $280C3
SAC4            EQU     $280C4
SAC5            EQU     $280C5
SAC6            EQU     $280C6
SAC7            EQU     $280C7
SAC8            EQU     $280C8
SAC9            EQU     $280C9
SACA            EQU     $280CA
SACB            EQU     $280CB
SACC            EQU     $280CC
SACD            EQU     $280CD
SACE            EQU     $280CE
SACF            EQU     $280CF
SAD0            EQU     $280D0
SAD1            EQU     $280D1
```

```
SAD2            EQU     $280D2
SAD3            EQU     $280D3
SAD4            EQU     $280D4
SAD5            EQU     $280D5
SAD6            EQU     $280D6
SAD7            EQU     $280D7
SAD8            EQU     $280D8
SAD9            EQU     $280D9
SADA            EQU     $280DA
SADB            EQU     $280DB
SADC            EQU     $280DC
SADD            EQU     $280DD
SADE            EQU     $280DE
SADF            EQU     $280DF
SAE0            EQU     $280E0
SAE1            EQU     $280E1
SAE2            EQU     $280E2
SAE3            EQU     $280E3
SAE4            EQU     $280E4
SAE5            EQU     $280E5
SAE6            EQU     $280E6
SAE7            EQU     $280E7
SAE8            EQU     $280E8
SAE9            EQU     $280E9
SAEA            EQU     $280EA
SAEB            EQU     $280EB
SAEC            EQU     $280EC
SAED            EQU     $280ED
SAEE            EQU     $280EE
SAEF            EQU     $280EF
SAF0            EQU     $280F0
SAF1            EQU     $280F1
SAF2            EQU     $280F2
SAF3            EQU     $280F3
SAF4            EQU     $280F4
SAF5            EQU     $280F5
SAF6            EQU     $280F6
SAF7            EQU     $280F7
SAF8            EQU     $280F8
SAF9            EQU     $280F9
SAFA            EQU     $280FA
SAFB            EQU     $280FB
SAFC            EQU     $280FC
SAFD            EQU     $280FD
SAFE            EQU     $280FE
SAFF            EQU     $280FF
;
SB00            EQU     $28100
SB10            EQU     $28101
SB20            EQU     $28102
SB30            EQU     $28103
SB40            EQU     $28104
SB50            EQU     $28105
SB60            EQU     $28106
SB70            EQU     $28107
SB80            EQU     $28108
SB90            EQU     $28109
```

```
SBA0            EQU     $2810A
SBB0            EQU     $2810B
SBC0            EQU     $2810C
SBD0            EQU     $2810D
SBE0            EQU     $2810E
SBF0            EQU     $2810F
SB01            EQU     $28110
SB11            EQU     $28111
SB21            EQU     $28112
SB31            EQU     $28113
SB41            EQU     $28114
SB51            EQU     $28115
SB61            EQU     $28116
SB71            EQU     $28117
SB81            EQU     $28118
SB91            EQU     $28119
SBA1            EQU     $2811A
SBB1            EQU     $2811B
SBC1            EQU     $2811C
SBD1            EQU     $2811D
SBE1            EQU     $2811E
SBF1            EQU     $2811F
SB02            EQU     $28120
SB12            EQU     $28121
SB22            EQU     $28122
SB32            EQU     $28123
SB42            EQU     $28124
SB52            EQU     $28125
SB62            EQU     $28126
SB72            EQU     $28127
SB82            EQU     $28128
SB92            EQU     $28129
SBA2            EQU     $2812A
SBB2            EQU     $2812B
SBC2            EQU     $2812C
SBD2            EQU     $2812D
SBE2            EQU     $2812E
SBF2            EQU     $2812F
SB03            EQU     $28130
SB13            EQU     $28131
SB23            EQU     $28132
SB33            EQU     $28133
SB43            EQU     $28134
SB53            EQU     $28135
SB63            EQU     $28136
SB73            EQU     $28137
SB83            EQU     $28138
SB93            EQU     $28139
SBA3            EQU     $2813A
SBB3            EQU     $2813B
SBC3            EQU     $2813C
SBD3            EQU     $2813D
SBE3            EQU     $2813E
SBF3            EQU     $2813F
SB04            EQU     $28140
SB14            EQU     $28141
SB24            EQU     $28142
```

```
SB34            EQU     $28143
SB44            EQU     $28144
SB54            EQU     $28145
SB64            EQU     $28146
SB74            EQU     $28147
SB84            EQU     $28148
SB94            EQU     $28149
SBA4            EQU     $2814A
SBB4            EQU     $2814B
SBC4            EQU     $2814C
SBD4            EQU     $2814D
SBE4            EQU     $2814E
SBF4            EQU     $2814F
SB05            EQU     $28150
SB15            EQU     $28151
SB25            EQU     $28152
SB35            EQU     $28153
SB45            EQU     $28154
SB55            EQU     $28155
SB65            EQU     $28156
SB75            EQU     $28157
SB85            EQU     $28158
SB95            EQU     $28159
SBA5            EQU     $2815A
SBB5            EQU     $2815B
SBC5            EQU     $2815C
SBD5            EQU     $2815D
SBE5            EQU     $2815E
SBF5            EQU     $2815F
SB06            EQU     $28160
SB16            EQU     $28161
SB26            EQU     $28162
SB36            EQU     $28163
SB46            EQU     $28164
SB56            EQU     $28165
SB66            EQU     $28166
SB76            EQU     $28167
SB86            EQU     $28168
SB96            EQU     $28169
SBA6            EQU     $2816A
SBB6            EQU     $2816B
SBC6            EQU     $2816C
SBD6            EQU     $2816D
SBE6            EQU     $2816E
SBF6            EQU     $2816F
SB07            EQU     $28170
SB17            EQU     $28171
SB27            EQU     $28172
SB37            EQU     $28173
SB47            EQU     $28174
SB57            EQU     $28175
SB67            EQU     $28176
SB77            EQU     $28177
SB87            EQU     $28178
SB97            EQU     $28179
SBA7            EQU     $2817A
SBB7            EQU     $2817B
```

```
SBC7            EQU     $2817C
SBD7            EQU     $2817D
SBE7            EQU     $2817E
SBF7            EQU     $2817F
SB08            EQU     $28180
SB18            EQU     $28181
SB28            EQU     $28182
SB38            EQU     $28183
SB48            EQU     $28184
SB58            EQU     $28185
SB68            EQU     $28186
SB78            EQU     $28187
SB88            EQU     $28188
SB98            EQU     $28189
SBA8            EQU     $2818A
SBB8            EQU     $2818B
SBC8            EQU     $2818C
SBD8            EQU     $2818D
SBE8            EQU     $2818E
SBF8            EQU     $2818F
SB09            EQU     $28190
SB19            EQU     $28191
SB29            EQU     $28192
SB39            EQU     $28193
SB49            EQU     $28194
SB59            EQU     $28195
SB69            EQU     $28196
SB79            EQU     $28197
SB89            EQU     $28198
SB99            EQU     $28199
SBA9            EQU     $2819A
SBB9            EQU     $2819B
SBC9            EQU     $2819C
SBD9            EQU     $2819D
SBE9            EQU     $2819E
SBF9            EQU     $2819F
SB0A            EQU     $281A0
SB1A            EQU     $281A1
SB2A            EQU     $281A2
SB3A            EQU     $281A3
SB4A            EQU     $281A4
SB5A            EQU     $281A5
SB6A            EQU     $281A6
SB7A            EQU     $281A7
SB8A            EQU     $281A8
SB9A            EQU     $281A9
SBAA            EQU     $281AA
SBBA            EQU     $281AB
SBCA            EQU     $281AC
SBDA            EQU     $281AD
SBEA            EQU     $281AE
SBFA            EQU     $281AF
SB0B            EQU     $281B0
SB1B            EQU     $281B1
SB2B            EQU     $281B2
SB3B            EQU     $281B3
SB4B            EQU     $281B4
```

```
SB5B            EQU     $281B5
SB6B            EQU     $281B6
SB7B            EQU     $281B7
SB8B            EQU     $281B8
SB9B            EQU     $281B9
SBAB            EQU     $281BA
SBBB            EQU     $281BB
SBCB            EQU     $281BC
SBDB            EQU     $281BD
SBEB            EQU     $281BE
SBFB            EQU     $281BF
SB0C            EQU     $281C0
SB1C            EQU     $281C1
SB2C            EQU     $281C2
SB3C            EQU     $281C3
SB4C            EQU     $281C4
SB5C            EQU     $281C5
SB6C            EQU     $281C6
SB7C            EQU     $281C7
SB8C            EQU     $281C8
SB9C            EQU     $281C9
SBAC            EQU     $281CA
SBBC            EQU     $281CB
SBCC            EQU     $281CC
SBDC            EQU     $281CD
SBEC            EQU     $281CE
SBFC            EQU     $281CF
SB0D            EQU     $281D0
SB1D            EQU     $281D1
SB2D            EQU     $281D2
SB3D            EQU     $281D3
SB4D            EQU     $281D4
SB5D            EQU     $281D5
SB6D            EQU     $281D6
SB7D            EQU     $281D7
SB8D            EQU     $281D8
SB9D            EQU     $281D9
SBAD            EQU     $281DA
SBBD            EQU     $281DB
SBCD            EQU     $281DC
SBDD            EQU     $281DD
SBED            EQU     $281DE
SBFD            EQU     $281DF
SB0E            EQU     $281E0
SB1E            EQU     $281E1
SB2E            EQU     $281E2
SB3E            EQU     $281E3
SB4E            EQU     $281E4
SB5E            EQU     $281E5
SB6E            EQU     $281E6
SB7E            EQU     $281E7
SB8E            EQU     $281E8
SB9E            EQU     $281E9
SBAE            EQU     $281EA
SBBE            EQU     $281EB
SBCE            EQU     $281EC
SBDE            EQU     $281ED
```

```
SBEE            EQU     $281EE
SBFE            EQU     $281EF
SB0F            EQU     $281F0
SB1F            EQU     $281F1
SB2F            EQU     $281F2
SB3F            EQU     $281F3
SB4F            EQU     $281F4
SB5F            EQU     $281F5
SB6F            EQU     $281F6
SB7F            EQU     $281F7
SB8F            EQU     $281F8
SB9F            EQU     $281F9
SBAF            EQU     $281FA
SBBF            EQU     $281FB
SBCF            EQU     $281FC
SBDF            EQU     $281FD
SBEF            EQU     $281FE
SBFF            EQU     $281FF
;
SC00            EQU     $28200
SC01            EQU     $28201
SC02            EQU     $28202
SC03            EQU     $28203
SC04            EQU     $28204
SC05            EQU     $28205
SC06            EQU     $28206
SC07            EQU     $28207
SC08            EQU     $28208
SC09            EQU     $28209
SC0A            EQU     $2820A
SC0B            EQU     $2820B
SC0C            EQU     $2820C
SC0D            EQU     $2820D
SC0E            EQU     $2820E
SC0F            EQU     $2820F
SC10            EQU     $28210
SC11            EQU     $28211
SC12            EQU     $28212
SC13            EQU     $28213
SC14            EQU     $28214
SC15            EQU     $28215
SC16            EQU     $28216
SC17            EQU     $28217
SC18            EQU     $28218
SC19            EQU     $28219
SC1A            EQU     $2821A
SC1B            EQU     $2821B
SC1C            EQU     $2821C
SC1D            EQU     $2821D
SC1E            EQU     $2821E
SC1F            EQU     $2821F
SC20            EQU     $28220
SC21            EQU     $28221
SC22            EQU     $28222
SC23            EQU     $28223
SC24            EQU     $28224
SC25            EQU     $28225
```

```
SC26            EQU     $28226
SC27            EQU     $28227
SC28            EQU     $28228
SC29            EQU     $28229
SC2A            EQU     $2822A
SC2B            EQU     $2822B
SC2C            EQU     $2822C
SC2D            EQU     $2822D
SC2E            EQU     $2822E
SC2F            EQU     $2822F
SC30            EQU     $28230
SC31            EQU     $28231
SC32            EQU     $28232
SC33            EQU     $28233
SC34            EQU     $28234
SC35            EQU     $28235
SC36            EQU     $28236
SC37            EQU     $28237
SC38            EQU     $28238
SC39            EQU     $28239
SC3A            EQU     $2823A
SC3B            EQU     $2823B
SC3C            EQU     $2823C
SC3D            EQU     $2823D
SC3E            EQU     $2823E
SC3F            EQU     $2823F
SC40            EQU     $28240
SC41            EQU     $28241
SC42            EQU     $28242
SC43            EQU     $28243
SC44            EQU     $28244
SC45            EQU     $28245
SC46            EQU     $28246
SC47            EQU     $28247
SC48            EQU     $28248
SC49            EQU     $28249
SC4A            EQU     $2824A
SC4B            EQU     $2824B
SC4C            EQU     $2824C
SC4D            EQU     $2824D
SC4E            EQU     $2824E
SC4F            EQU     $2824F
SC50            EQU     $28250
SC51            EQU     $28251
SC52            EQU     $28252
SC53            EQU     $28253
SC54            EQU     $28254
SC55            EQU     $28255
SC56            EQU     $28256
SC57            EQU     $28257
SC58            EQU     $28258
SC59            EQU     $28259
SC5A            EQU     $2825A
SC5B            EQU     $2825B
SC5C            EQU     $2825C
SC5D            EQU     $2825D
SC5E            EQU     $2825E
```

```
SC5F            EQU     $2825F
SC60            EQU     $28260
SC61            EQU     $28261
SC62            EQU     $28262
SC63            EQU     $28263
SC64            EQU     $28264
SC65            EQU     $28265
SC66            EQU     $28266
SC67            EQU     $28267
SC68            EQU     $28268
SC69            EQU     $28269
SC6A            EQU     $2826A
SC6B            EQU     $2826B
SC6C            EQU     $2826C
SC6D            EQU     $2826D
SC6E            EQU     $2826E
SC6F            EQU     $2826F
SC70            EQU     $28270
SC71            EQU     $28271
SC72            EQU     $28272
SC73            EQU     $28273
SC74            EQU     $28274
SC75            EQU     $28275
SC76            EQU     $28276
SC77            EQU     $28277
SC78            EQU     $28278
SC79            EQU     $28279
SC7A            EQU     $2827A
SC7B            EQU     $2827B
SC7C            EQU     $2827C
SC7D            EQU     $2827D
SC7E            EQU     $2827E
SC7F            EQU     $2827F
SC80            EQU     $28280
SC81            EQU     $28281
SC82            EQU     $28282
SC83            EQU     $28283
SC84            EQU     $28284
SC85            EQU     $28285
SC86            EQU     $28286
SC87            EQU     $28287
SC88            EQU     $28288
SC89            EQU     $28289
SC8A            EQU     $2828A
SC8B            EQU     $2828B
SC8C            EQU     $2828C
SC8D            EQU     $2828D
SC8E            EQU     $2828E
SC8F            EQU     $2828F
SC90            EQU     $28290
SC91            EQU     $28291
SC92            EQU     $28292
SC93            EQU     $28293
SC94            EQU     $28294
SC95            EQU     $28295
SC96            EQU     $28296
SC97            EQU     $28297
```

```
SC98            EQU     $28298
SC99            EQU     $28299
SC9A            EQU     $2829A
SC9B            EQU     $2829B
SC9C            EQU     $2829C
SC9D            EQU     $2829D
SC9E            EQU     $2829E
SC9F            EQU     $2829F
SCA0            EQU     $282A0
SCA1            EQU     $282A1
SCA2            EQU     $282A2
SCA3            EQU     $282A3
SCA4            EQU     $282A4
SCA5            EQU     $282A5
SCA6            EQU     $282A6
SCA7            EQU     $282A7
SCA8            EQU     $282A8
SCA9            EQU     $282A9
SCAA            EQU     $282AA
SCAB            EQU     $282AB
SCAC            EQU     $282AC
SCAD            EQU     $282AD
SCAE            EQU     $282AE
SCAF            EQU     $282AF
SCB0            EQU     $282B0
SCB1            EQU     $282B1
SCB2            EQU     $282B2
SCB3            EQU     $282B3
SCB4            EQU     $282B4
SCB5            EQU     $282B5
SCB6            EQU     $282B6
SCB7            EQU     $282B7
SCB8            EQU     $282B8
SCB9            EQU     $282B9
SCBA            EQU     $282BA
SCBB            EQU     $282BB
SCBC            EQU     $282BC
SCBD            EQU     $282BD
SCBE            EQU     $282BE
SCBF            EQU     $282BF
SCC0            EQU     $282C0
SCC1            EQU     $282C1
SCC2            EQU     $282C2
SCC3            EQU     $282C3
SCC4            EQU     $282C4
SCC5            EQU     $282C5
SCC6            EQU     $282C6
SCC7            EQU     $282C7
SCC8            EQU     $282C8
SCC9            EQU     $282C9
SCCA            EQU     $282CA
SCCB            EQU     $282CB
SCCC            EQU     $282CC
SCCD            EQU     $282CD
SCCE            EQU     $282CE
SCCF            EQU     $282CF
SCD0            EQU     $282D0
```

```
SCD1            EQU     $282D1
SCD2            EQU     $282D2
SCD3            EQU     $282D3
SCD4            EQU     $282D4
SCD5            EQU     $282D5
SCD6            EQU     $282D6
SCD7            EQU     $282D7
SCD8            EQU     $282D8
SCD9            EQU     $282D9
SCDA            EQU     $282DA
SCDB            EQU     $282DB
SCDC            EQU     $282DC
SCDD            EQU     $282DD
SCDE            EQU     $282DE
SCDF            EQU     $282DF
SCE0            EQU     $282E0
SCE1            EQU     $282E1
SCE2            EQU     $282E2
SCE3            EQU     $282E3
SCE4            EQU     $282E4
SCE5            EQU     $282E5
SCE6            EQU     $282E6
SCE7            EQU     $282E7
SCE8            EQU     $282E8
SCE9            EQU     $282E9
SCEA            EQU     $282EA
SCEB            EQU     $282EB
SCEC            EQU     $282EC
SCED            EQU     $282ED
SCEE            EQU     $282EE
SCEF            EQU     $282EF
SCF0            EQU     $282F0
SCF1            EQU     $282F1
SCF2            EQU     $282F2
SCF3            EQU     $282F3
SCF4            EQU     $282F4
SCF5            EQU     $282F5
SCF6            EQU     $282F6
SCF7            EQU     $282F7
SCF8            EQU     $282F8
SCF9            EQU     $282F9
SCFA            EQU     $282FA
SCFB            EQU     $282FB
SCFC            EQU     $282FC
SCFD            EQU     $282FD
SCFE            EQU     $282FE
SCFF            EQU     $282FF
;
; Semaphores
SML1A           EQU     $28300
SML2A           EQU     $28301
SML1B           EQU     $28302
SML2B           EQU     $28303
SMLC            EQU     $28304
SM1S            EQU     $28305
SM2F            EQU     $28306
SM2S            EQU     $28307
```

```
;
LCW             EQU     $30003
WC1LB           EQU     $30000
WC1MB           EQU     $30000
RC1LB           EQU     $30000
RC1MB           EQU     $30000
GtRd            EQU     $8000
GtRda           EQU     $38000
ICNT            EQU     $17000
RCNT            EQU     $17004
;
;
; Matrix Control Equates
;       Byte equates
ACNT            EQU     $17010
MMWL            EQU     $17011
MMTA            EQU     $17012
LMASB           EQU     $17013
LMBSB           EQU     $17014
ZERO            EQU     $17015
PROCa           EQU     $17016
MMTSB           EQU     $17017
MMTSC           EQU     $17018
PrBa            EQU     $17019
PrBb            EQU     $17020
MMTB            EQU     $17021
;       Word equates
BCNT            EQU     $17040
BSCNT           EQU     $17042
;       Long Equates
MCSVB           EQU     $17050
LMAS            EQU     $17054
LMBS            EQU     $17058
ACRT            EQU     $1705C
BCRT            EQU     $17060
CCRT            EQU     $17064
ACRTa           EQU     $17068
BCRTa           EQU     $1706C
CCRTa           EQU     $17070
ACRTb           EQU     $17074
BCRTb           EQU     $17078
CCRTb           EQU     $1707C
ACRTc           EQU     $17080
BCRTc           EQU     $17084
CCRTc           EQU     $17088
SD5             EQU     $1708C
SD6             EQU     $17090
; Matrix A Load Values
LMAVA           EQU     $1708C
LMAVB           EQU     $17090
; Matrix B Load Values
LMBVA           EQU     $17070
LMBVB           EQU     $17074
;ASCNT
;CSCNT
;CCNT
;
```

```
                ORG     $10000
; Step A1
; Clearing of Registers
;
START           clr.l   D0
                clr.l   D1
                clr.l   D2
                clr.l   D3
                clr.l   D4
                clr.l   D5
                sub.l   A1,A1
                sub.l   A2,A2
                sub.l   A3,A3
                sub.l   A4,A4
                sub.l   A5,A5
;
;
; Routine for clearing $14000-$140C0,$28000-$280C0,$28100-$28106
;
                move.l  #$17000,A5
                move.l  #$17100,A4
                BSR     MCLR
                move.b  #PROC,(PROCa)
                cmpi.b  #$01,(PROCa)
                beq     MC0
                move.l  #$28000,A5
                move.l  #$28300,A4
                BSR     MCLR
                move.l  #$28300,A5
                move.l  #$28308,A4
                BSR     MCLR
MC0             move.l  #$14000,A5
                move.l  #$14300,A4
                BSR     MCLR
;
; Matrix Load variable
                move.l  #ASRT,(LMAVA)
                move.l  #ASRT,(LMAVB)
                move.l  #ASRT,(LMAS)
                move.l  #BSRT,(LMBVA)
                move.l  #BSRT,(LMBVB)
                move.l  #BSRT,(LMBS)
                move.b  #LMBVAL,D0
                move.b  #$00,(ZERO)
                move.b  #MMTS,(MMTSC)
                move.b  #MMTSA,(MMTSB)
                cmpi.b  #$01,(MMTSB)
                beq     LVA
                move.b  #MMT,(MMTA)
                move.b  #MMT,(MMTB)
                bra     LVB
LVA             cmpi.b  #$04,(MMTA)
                beq     LVA1
                move.b  #$01,(MMTA)
                bra     LVA2
LVA1            move.b  #$02,(MMTA)
LVA2            move.b  #$01,(MMTB)
```

```
LVB             cmpi.b  #$01,(MMTA)
                beq     LM4
                cmpi.b  #$02,(MMTA)
                beq     LM8
                cmpi.b  #$04,(MMTA)
                beq     LM16
LM4             cmpi.b  #$01,(MMTB)
                beq     LM4b
                addi.l  #$04,(LMAS)
                addi.l  #$04,(LMBS)
                move.b  #$04,(LMASB)
                move.b  #$04,(LMBSB)
                cmpi.b  #$01,(MMTSC)
                beq     LM4a
                addi.l  #$40,(LMAVB)
                move.w  #$0010,(BSCNT)
                BRA     LM4a3
LM4a            cmpi.b  #$01,(PROCa)
                beq     LM4a1
                addi.l  #$20,(LMAVB)
                bra     LM4a2
LM4a1           addi.l  #$20,(LMAVA)
                addi.l  #$20,(LMAS)
                addi.l  #$40,(LMAVB)
LM4a2           move.w  #$0010,(BSCNT)
LM4a3           addi.l  #$40,(LMBVB)
                move.b  #$20,(PrBb)
                BRA     LMO
LM4b            cmpi.b  #$01,(PROCa)
                beq     LM4b1
                addi.l  #$08,(LMAS)
                addi.l  #$04,(LMBS)
                move.b  #$08,(LMASB)
                move.b  #$04,(LMBSB)
                addi.l  #$40,(LMAVB)
                addi.l  #$80,(LMBVB)
                bra     LM4d
LM4b1           addi.l  #$08,(LMAS)
                addi.l  #$08,(LMBS)
                move.b  #$08,(LMASB)
                move.b  #$08,(LMBSB)
                addi.l  #$40,(LMAVA)
                addi.l  #$04,(LMBVA)
                addi.l  #$40,(LMAS)
                addi.l  #$04,(LMBS)
                addi.l  #$80,(LMAVB)
                addi.l  #$84,(LMBVB)
                move.b  #LMBVla,D0
LM4d            move.b  #$04,(PrBa)
                move.b  #$40,(PrBb)
                move.w  #$0010,(BSCNT)
                bra     LMO
LM8             cmpi.b  #$01,(MMTB)
                beq     LM8b
                addi.l  #$08,(LMAS)
                addi.l  #$08,(LMBS)
                move.b  #$08,(LMASB)
```

```
                move.b  #$08,(LMBSB)
                cmpi.b  #$01,(MMTSC)
                beq     LM8a
                addi.l  #$80,(LMAVB)
                move.w  #$0040,(BSCNT)
                BRA     LM8a3
LM8a            cmpi.b  #$01,(PROCa)
                beq     LM8a1
                addi.l  #$40,(LMAVB)
                bra     LM8a2
LM8a1           addi.l  #$40,(LMAVA)
                addi.l  #$40,(LMAS)
                addi.l  #$80,(LMAVB)
LM8a2           move.w  #$0020,(BSCNT)
LM8a3           addi.l  #$80,(LMBVB)
                move.b  #$40,(PrBb)
                BRA     LMO
LM8b            cmpi.b  #$01,(PROCa)
                beq     LM8b1
                addi.l  #$10,(LMAS)
                addi.l  #$08,(LMBS)
                move.b  #$10,(LMASB)
                move.b  #$08,(LMBSB)
                addi.l  #$80,(LMAVB)
                addi.l  #$100,(LMBVB)
                bra     LM8d
LM8b1           addi.l  #$10,(LMAS)
                addi.l  #$10,(LMBS)
                move.b  #$10,(LMASB)
                move.b  #$10,(LMBSB)
                addi.l  #$80,(LMAVA)
                addi.l  #$08,(LMBVA)
                addi.l  #$80,(LMAS)
                addi.l  #$08,(LMBS)
                addi.l  #$80,(LMAVB)
                addi.l  #$108,(LMBVB)
                move.b  #LMBVla,D0
LM8d            move.b  #$08,(PrBa)
                move.b  #$80,(PrBb)
                move.w  #$0020,(BSCNT)
                bra     LMO
;LM8             addi.l  #$08,(LMAS)
;               addi.l  #$08,(LMBS)
;               move.b  #$08,(LMASB)
;               move.b  #$08,(LMBSB)
;               addi.l  #$80,(LMBVB)
;               cmpi.b  #MMTS,(ZERO)
;               bne     S2
;               move.w  #$0040,(BSCNT)
;               addi.l  #$80,(LMAVB)
;               BRA     S3
;S2              move.w  #$0020,(BSCNT)
;               cmpi.b  #$01,(PROCa)
;               beq     S2a
;               addi.l  #$40,(LMAVB)
;               bra     LMO
;S2a             move.b  #$40,(PrBb)
```

```
;                addi.l  #$40,(LMAVA)
;                addi.l  #$40,(LMAS)
;                addi.b  #$40,(LMASB)
;                addi.l  #$80,(LMAVB)
;S3              bra     LMO
LM16            addi.l  #$00,(LMAS)
                addi.l  #$00,(LMBS)
                addi.b  #$00,(LMASB)
                addi.b  #$00,(LMBSB)
                addi.l  #$100,(LMBVB)
                cmpi.b  #MMTS,(ZERO)
                bne     S4
                move.w  #$0100,(BSCNT)
                addi.l  #$100,(LMAVB)
                move.b  #$80,(PrBb)
                BRA     LMO
S4              move.w  #$0080,(BSCNT)
                cmpi.b  #$01,(PROCa)
                beq     S4a
                addi.l  #$80,(LMAVB)
                move.b  #$80,(PrBb)
                bra     LMO
S4a             addi.l  #$80,(LMAVA)
                addi.l  #$100,(LMAVB)
                move.b  #$80,(PrBb)
; Loading Variables
LMO             move.b  #$00,(ACNT)
                move.w  #$00,(BCNT)
                move.b  #$00,(MMWL)
                move.l  #$00,(ACRT)
                move.l  #$00,(BCRT)
                move.l  #$00,(CCRT)
; Initializing the Counter
                move.b  #$30,(LCW)
;
; Loading of Matrix Value
;                BSR     Lmat    ; Testing
; Matrix A
                movea.l (LMAVA),A3
                movea.l (LMAVB),A2
                BSR     LMA
;
; Matrix B
                movea.l (LMBVA),A3
                movea.l (LMBVB),A2
                BSR     LMB
;
; Check for Single Processor
                cmpi.b  #$01,(MMTSC)
                bne     SG
;
; Check if Prog A or Prog B
                cmpi.b  #$01,(PROCa)
                beq     ASWT
; Locking Semaphores
                BSR     LS
;
```

```
; Unlocking Semaphores
                BSR     US
;
; Waiting for PB initialization
                movea.l #SM1S,A3        ; PA Only
                BSR     SC
;
; Routine to Start Time
SG              BSR     TSTR            ; PA only
;
; Starting Processor B
                clr.b  (SM2S)        ; PA only
                bra    SG1
;
; PB Initialization
ASWT            clr.b   (SM1S)          ;PB only
; Processor B Start
                movea.l #SM2S,A3       ;PB Only
                BSR     SC             ;PB Only
;
; Matrix Multiplication
;
; [Segment A0]
;
SG1              move.l  #ASRT,(ACRT)
                move.l  #BSRT,(BCRT)
                move.l  #CSRT,(CCRT)
                clr.l   D5
                clr.l   D6
                move.b  (PrBa),D5
                move.b  (PrBb),D6
                cmpi.b  #$01,(PROCa)
                bne     SA1
; Start Locations for Proc B
                add.l   D6,(ACRT)
                cmpi.b  #$01,(MMTSB)
                bne     PB0
                add.l   D5,(BCRT)
PB0             add.l   D6,(CCRT)
                move.l  (ACRT),(ACRTb)
                move.l  (BCRT),(BCRTb)
                move.l  (CCRT),(CCRTb)
                bra     SA2
; Location Start for Block Proc A
;
SA1             move.l  (ACRT),(ACRTa)
                move.l  (BCRT),(BCRTa)
                move.l  (CCRT),(CCRTa)
SA2             cmpi.b  #$01,(MMTSB)
                bne     PA
                cmpi.b  #$01,(PROCa)
                bne     PA
                movea.l (BCRT),A5
                BSR     MTSM
                clr.b   (SML1A)
                movea.l (BCRT),A5
                adda.l  D6,A5
```

```
                BSR     MTSM
                clr.b   (SML2A)
                add.l   D5,(ACRTb)
                movea.l (ACRTb),A4
                movea.l (BCRT),A5
                movea.l (CCRT),A3

                BSR     Block
                movea.l (ACRTb),A4
                movea.l (BCRT),A5
                movea.l (CCRT),A3
                adda.l  D6,A5
                adda.l  D5,A3
                BSR     Block
                sub.l   D5,(BCRTb)
                movea.l (BCRTb),A5
;
                movea.l #SML1B,A3
                BSR     SC
;
                BSR     GFSM
                movea.l (ACRT),A4
                movea.l (BCRTb),A5
                movea.l (CCRT),A3
                BSR     Block
                add.l   D6,(BCRTb)
                movea.l (BCRTb),A5
;
                movea.l #SML2B,A3
                BSR     SC
;
                BSR     GFSM
                movea.l (ACRT),A4
                movea.l (BCRTb),A5
                movea.l (CCRT),A3
                adda.l  D5,A3
                BSR     Block
                bra     MMC
PA              cmpi.b  #$01,(PROCa)
                beq     PA0
                movea.l (ACRTa),A4
                movea.l (BCRTa),A5
                movea.l (CCRTa),A3
                bra     PA1
PA0             movea.l (ACRTb),A4
                movea.l (BCRTb),A5
                movea.l (CCRTb),A3
PA1             BSR Block
                cmpi.b  #$01,(MMTSB)
                bne     MMC
                movea.l (ACRT),A4
                movea.l (BCRT),A5
                movea.l (CCRT),A3
                adda.l  D6,A5
                adda.l  D5,A3
                BSR     Block
                movea.l (BCRT),A5
```

```
                BSR     MTSM
                clr.b   (SML1B)
                move.l (BCRT),(BCRTa)
                movea.l (BCRTa),A5
                adda.l  D5,A5
;
                movea.l #SML1A,A3
                BSR     SC
;
                BSR     GFSM
                movea.l (ACRT),A4
                movea.l (BCRTa),A5
                movea.l (CCRT),A3
                adda.l  D5,A4
                adda.l  D5,A5
                BSR     Block
                movea.l (BCRT),A5
                adda.l  D6,A5
                BSR     MTSM
                clr.b   (SML2B)
                movea.l (BCRTa),A5
                adda.l  D6,A5
                adda.l  D5,A5
;
                movea.l #SML2A,A3
                BSR     SC
;
                BSR     GFSM
                movea.l (ACRT),A4
                movea.l (BCRTa),A5
                movea.l (CCRT),A3
                adda.l  D5,A4
                adda.l  D6,A5
                adda.l  D5,A5
                adda.l  D5,A3
                BSR     Block
;
;
; [Segment A9]
;
; Checking SMLC Semaphore
MMC             movea.l #SMLC,A3
                BSR     SC
;
; Move Results to Shared Memory
                movea.l #CSRT,A5
                BSR     MRTSM
;
; Clearing SMLC Semaphore
                clr.b   (SMLC)
;
; [Segment A10]
;
                cmpi.b  #$01,(PROCa)
                beq     DSWT
                cmpi.b  #$01,(MMTSC)
                bne     CSWT
```

```
; Checking if Processor 2 is finished
                movea.l #SM2F,A3
                BSR     SC
;
; Routine to Stop Timer
CSWT            BSR     TSTP
;
                bra     ESWT
;
; Processor 2 is finsished
DSWT            clr.b (SM2F)
; Routine to Get Time information
ESWT            BRA     ENDING
;
;
; Subroutines
TSTR            move.b  #$00,(WC1LB)
                move.b  #$00,(WC1MB)
                move.b  #$00,(GtRd)
                RTS
TSTPA           move.b  #$01,(GtRd)
                move.b  #$03,(GtRda)
                RTS
TSTP            move.b  #$01,(GtRd)
                move.b  #$00,(LCW)
                clr.l   D1
                clr.l   D2
                move.b  (RC1LB),D1
                move.b  (RC1MB),D2
                ASL     #$8,D2
                add.l   D2,D1
                move.l  D1,(RCNT)
                RTS
;
;SC               subi.b  #$01,D7
;                cmpi.b  #$00,D7
;                bne     SC
SC              TAS     (A3)
                BNE     SC
;                clr.b   (A3)
                RTS
;
BLOCK           move.l  D5,(SD5)
                move.l  D6,(SD6)
                movea.w (BSCNT),A2
                move.l  A4,(ACRTc)
                move.l  A5,(BCRTc)
                move.l  A3,(CCRTc)
BLA             BSR     MMW
                addi.b  #$01,(MMWL)
                move.b  (MMTB),D3
                cmp.b   (MMWL),D3
                bne     BLA
BLB             addi.w  #$01,BCNT
                cmpa.w  (BCNT),A2
                beq     BLEND
;                move.w  #$01,CCNT
```

```
                addi.b  #$01,ACNT
                adda.l  #$01,A3
                cmpi.b  #$01,(MMWL)
                beq     MMW4
                cmpi.b  #$02,(MMWL)
                beq     MMW8
MMW16           move.b  #$00,MMWL
                cmpi.b  #$10,(ACNT)
                beq     ADJA16
                BRA     BLCNT
MMW8            move.b  #$00,MMWL
                cmpi.b  #$08,(ACNT)
                beq     ADJA08
                adda.l  #$08,A5
                BRA     BLCNT
MMW4            move.b  #$00,MMWL
                cmpi.b  #$04,(ACNT)
                beq     ADJA04
                adda.l  #$0C,A5
                BRA     BLCNT
ADJA16          addi.l  #$10,(ACRTc)
                movea.l (BCRTc),A5
                move.b  #$00,(ACNT)
                BRA     BLCNT
ADJA08          addi.l  #$10,(ACRTc)
                addi.l  #$10,(CCRTc)
                movea.l (BCRTc),A5
                movea.l (CCRTc),A3
                move.b  #$00,(ACNT)
                BRA     BLCNT
ADJA04          addi.l  #$10,(ACRTc)
                addi.l  #$10,(CCRTc)
                movea.l (ACRTc),A4
                movea.l (BCRTc),A5
                movea.l (CCRTc),A3
                move.b  #$00,(ACNT)
                BRA     BLCNT
BLCNT           movea.l (ACRTc),A4
;                move.w  #$01,CCNT
                BRA     BLA
BLEND           move.b  #$00,MMWL
                move.b  #$00,ACNT
                move.w  #$00,BCNT
                move.l  (SD5),D5
                move.l  (SD6),D6
                RTS
;
MMW             move.b  (A4)+,D0
                move.b  (A5)+,D1
                move.b  (A4)+,D2
                move.b  (A5)+,D3
                move.b  (A4)+,D4
                move.b  (A5)+,D5
                move.b  (A4)+,D6
                move.b  (A5)+,D7
                mulu    D1,D0
                mulu    D3,D2
```

```
                add.b   D0,D2
                mulu    D5,D4
                mulu    D7,D6
                add.b   D4,D6
                add.b   D6,D2
                clr.l   D5
                move.b  (A3),D5
                add.b   D5,D2
                move.b  D2,(A3)
                clr.l   D1
                clr.l   D2
                clr.l   D3
                clr.l   D4
                clr.l   D5
                clr.l   D6
                clr.l   D7
;                sub.l   A3,A3
                RTS
;
MTSM            clr.l   D0
                clr.l   D1
                clr.l   D2
                move.b  (PrBa),D1
                move.b  (PrBb),D2
                movea.l A5,A2
                movea.l A5,A3
                adda.l  D1,A2
                adda.l  D2,A3
                move.l  A5,D0
                addi.l  #$14000,D0
                move.l  D0,A1
MT1             move.b  (A5)+,(A1)+
                cmpa.l  A5,A2
                bne     MT2
                adda.l  #$10,A2
                adda.l  #$0C,A1
                adda.l  #$0C,A5
MT2             cmpa.l  A5,A3
                bne     MT1
                sub.l   A1,A1
                sub.l   A2,A2
                sub.l   A3,A3
                sub.l   A5,A5
                RTS
;
GFSM            clr.l   D0
                clr.l   D1
                clr.l   D2
                move.b  (PrBa),D1
                move.b  (PrBb),D2
                movea.l A5,A2
                movea.l A5,A3
                adda.l  D1,A2
                adda.l  D2,A3
                move.l  A5,D0
                addi.l  #$14000,D0
                move.l  D0,A1
```

```
MT4             move.b  (A1)+,(A5)+
                cmpa.l  A5,A2
                bne     MT5
                adda.l  #$10,A2
                adda.l  #$0C,A1
                adda.l  #$0C,A5
MT5             cmpa.l  A5,A3
                bne     MT4
                sub.l   A1,A1
                sub.l   A2,A2
                sub.l   A3,A3
                sub.l   A5,A5
                RTS
;
MRTSM           clr.l   D1
                movea.l A5,A2
                movea.l A5,A3
                cmpi.b  #$01,(MMTA)
                beq     MT5a
                cmpi.b  #$02,(MMTA)
                beq     MT5b
                cmpi.b  #$04,(MMTA)
                beq     MT5c
MT5a            cmpi.b  #$01,(MMTB)
                beq     MT5a5
                cmpi.b  #$01,(MMTSC)
                beq     MT5a2
                adda.l  #$40,A3
                bra     MT5a4
MT5a2           cmpi.b  #$01,(PROCa)
                beq     MT5a3
                adda.l  #$20,A3
                bra     MT5a4
MT5a3           adda.l  #$20,A5
                adda.l  #$40,A3
MT5a4           adda.l  #$04,A2
                move.l  #$04,D1
                bra     MT5d
MT5a5           cmpi.b  #$01,(PROCa)
                beq     MT5a6
                adda.l  #$40,A3
                bra     MT5a7
MT5a6           adda.l  #$40,A5
                adda.l  #$40,A2
                adda.l  #$80,A3
MT5a7           adda.l  #$08,A2
                move.l  #$08,D1
                bra     MT5d
MT5b            cmpi.b  #$01,(MMTB)
                beq     MT5b5
                cmpi.b  #$01,(MMTSC)
                beq     MT5b2
                adda.l  #$80,A3
                bra     MT5b4
MT5b2           cmpi.b  #$01,(PROCa)
                beq     MT5b3
                adda.l  #$40,A3
```

```
                bra     MT5b4
MT5b3           adda.l  #$40,A5
                adda.l  #$80,A3
MT5b4           adda.l  #$08,A2
                move.l  #$08,D1
                bra     MT5d
MT5b5           cmpi.b  #$01,(PROCa)
                beq     MT5b6
                adda.l  #$80,A3
                bra     MT5b7
MT5b6           adda.l  #$80,A5
                adda.l  #$80,A2
                adda.l  #$100,A3
MT5b7           adda.l  #$10,A2
                move.l  #$10,D1
                bra     MT5d
MT5c            cmpi.b  #$01,(MMTSC)
                beq     MT5c2
                adda.l  #$100,A3
                bra     MT5c4
MT5c2           cmpi.b  #$01,(PROCa)
                beq     MT5c3
                adda.l  #$80,A3
                bra     MT5c4
MT5c3           adda.l  #$80,A5
                adda.l  #$100,A3
MT5c4           adda.l  #$10,A2
                move.l  #$10,D1
;
MT5d            move.l  A5,D0
                addi.l  #$14000,D0
                move.l  D0,A1
MT6             move.b  (A5)+,(A1)+
                cmpa.l  A5,A2
                bne     MT7
                cmpi.b  #$04,(MMTA)
                beq     MT7
                adda.l  #$10,A2
                adda.l  D1,A1
                adda.l  D1,A5
MT7             cmpa.l  A5,A3
                bne     MT6
                sub.l   A1,A1
                sub.l   A2,A2
                sub.l   A3,A3
                sub.l   A5,A5
                RTS
;
;
MCLR            clr.b   (A5)+
                cmpa.l  A4,A5
                BNE     MCLR
                sub.l   A5,A5
                sub.l   A4,A4
                RTS
;
LS              move.b #$80,(SML1A)  ; PA only
```

```
                move.b #$80,(SML2A) ; PA only
                move.b #$80,(SML1B) ; PA only
                move.b #$80,(SML2B) ; PA only
                move.b #$80,(SM1S)  ; PA only
                move.b #$80,(SM2F)  ; PA only
                move.b #$80,(SM2S)  ; PA only
                RTS
;
US              move.b #$00,(SMLC)  ; PA only
                RTS
;
LMA             cmpa.l  (LMAS),A3
                bne     LMAC
                cmpi.b  #$08,(LMASB)
                beq     LMAA
                cmpi.b  #$04,(LMASB)
                beq     LMAB
                BRA     LMAC
LMAA            adda.l  #$08,A3
                addi.l  #$10,(LMAS)
                BRA     LMAC
LMAB            adda.l  #$0C,A3
                addi.l  #$10,(LMAS)
LMAC            cmpa.l  A2,A3
                beq     LMAE
                move.b  #LMAVal,(A3)+
                BRA     LMA
LMAE            RTS
;
LMB             cmpa.l  (LMBS),A3
                bne     LMBC
                cmpi.b  #$08,(LMBSB)
                beq     LMBA
                cmpi.b  #$04,(LMBSB)
                beq     LMBB
                BRA     LMBC
LMBA            adda.l  #$08,A3
                addi.l  #$10,(LMBS)
                BRA     LMBC
LMBB            adda.l  #$0C,A3
                addi.l  #$10,(LMBS)
LMBC            cmpa.l  A2,A3
                beq     LMBE
                move.b  D0,(A3)+
                BRA     LMB
LMBE            RTS
;
;
ENDING          Trap    #9
                END     START
```

REFERENCES

Rosenstark, Dr. Sol, 1998, *Computer Construction Project and Experiments – EE 393: Electrical Engineering Laboratory III.* New Jersey: New Jersey Institute of Technology.

*******The following references have been included as further reading to help with*******
the understanding of the material, figures, and programs contained in this thesis.

Hwang, Kai, 1993, *ADVANCED COMPUTER ARCHITECTURE: Parallelism, Scalability, Programmability.* New York: McGraw-Hill, Inc.

Antonakos, James L., 1996, *THE 68000 MICROPROCESSOR: Hardware and Software Principles and Applications.* Third Edition. New Jersey: Prentice Hall.

Anon., 1992, *M68000: Family Programmer's Reference Manual.* Arizona: Motorola Literature Distribution.

Katz, Randy H., 1994, *CONTEMPORARY LOGIC DESIGN.* California: The Benjamin/Cummings Publishing Company, Inc.