ABSTRACT

Title of Thesis:	Extended Split-Issue Mechanism in VLIW DSPs to Support SMT and Hardware-ISA Decoupling
Degree candidate:	Bharath Iyer
Degree and year:	Master of Science, 2003
Thesis directed by:	Professor Bruce Jacob Department of Electrical Engineering

The use of VLIW architecture has become widespread in the DSP arena due to the combined benefits of simple hardware and instruction level parallelism extracted by the compiler. In an attempt to extract higher processing throughput we investigate simultaneous multithreading (SMT) to exploit thread-level parallelism in a VLIW DSP. We propose an extended split-issue mechanism that allows us to break the instruction packet syntax of a VLIW compiler without violating the dataflow dependences. The capability to break VLIW instruction packets gives us greater freedom in scheduling instructions from different threads based on functional unit availability. We find that the SMT VLIW DSP provides significant processing throughput gains, which could also be traded-off for rich energy savings. The extended split-issue mechanism

also provides us with an unprecedented, albeit limited, degree of freedom in disassociating the hardware implementation of the VLIW processor with the instruction set architecture (ISA). Simulation results show that the ability to modify the hardware implementation based on the utilisation statistics presents a new dimension that can be exploited to improve the efficiency of the processor.

Extended Split-Issue Mechanism in VLIW DSPs to Support SMT and Hardware-ISA Decoupling

by

Bharath Iyer

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2003

Advisory Committee:

Professor Bruce Jacob, Chair
Professor Manoj Franklin
Professor Donald Yeung

DEDICATION

To my parents Komala and Chandran...

who gave me life.

ACKNOWLEDGMENTS

Here's a big *Thank You* to all those people who contributed to this endeavour in different ways.

First and foremost, I believe one can never really thank one's *guru* (advisor). I shall forever be indebted to Prof. Bruce Jacob for insightful conversations, his sagacious obiter dicta and for providing me with the SCA lab facilities.

Sadagopan has been a great consociate with immense patience and ability to tolerate my fastidiousness. His inquisitive nature and superlative "web-browsing" skills have been a valuable asset to our *team*.

I wish to extend my gratitude to the thesis committee, comprising of Prof. Manoj Franklin and Prof. Donald Yeung, for taking time off their busy schedule. Their encouraging comments have helped shape some aspects of this thesis.

Special thanks is due to Paul Kohout, who built an excellent software simulator for the TMS320C6201 VLIW DSP.

For more than a year now the SCA lab has been my home-away-from-home. The SCALions and non-SCALions who made it a great place to live in are Aamer, Ankush, Brinda, Dave, Gautham, Himabindu, Kursad, Lei and Mohamed. All the technical and non-technical confabulations with these great

people have contributed towards my better understanding of Zen and the art of Computer Architecture.

Ofcourse, my days would have been incomplete without the regular tea sessions and appreciation is due for those who dared to be a part of my tea-club.

I express my humble gratitude to my parents, brother and near-and-dears for their steadfast and everlasting support and love.

Bharath Iyer

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

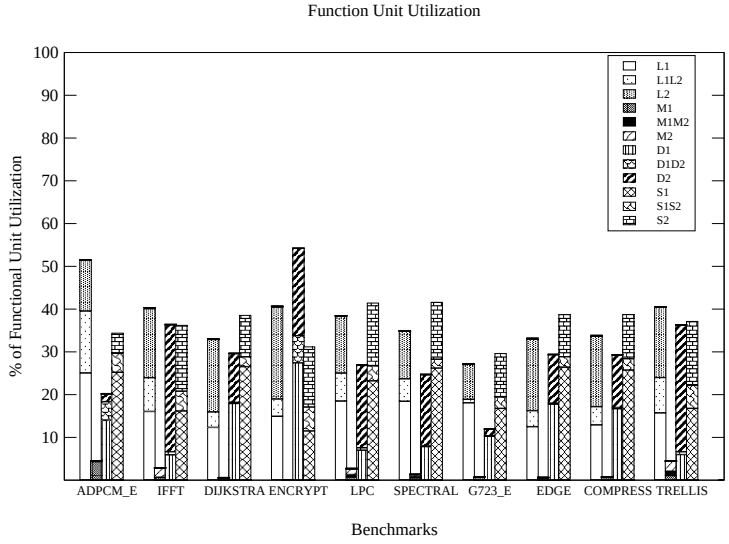
Introduction

1.1 VLIW DSPs

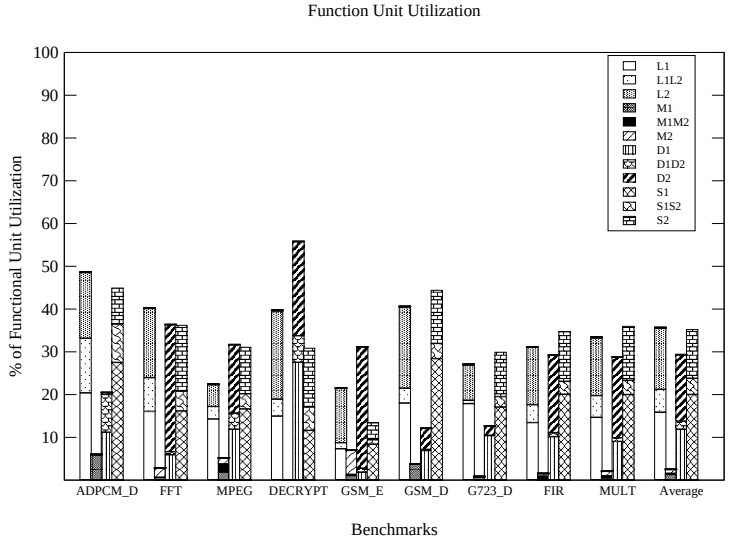
Since most DSP applications are embedded, the use of time-consuming assembly language programming was once justified as it resulted in more efficient use of hardware resources. With the software development cycle having an ever-increasing impact on time-to-market on product development efforts, more and more DSP programmers turned towards high-level languages, like C, to speed-up the development cycle. However, traditional DSP instruction sets are known to be rigid and extremely specialised making it very hard for a compiler to generate efficient code [1]. It was then observed that the regular algorithmic structure and dataflow properties of DSP applications made them well suited for VLIW architectures [2]. VLIW offered the advantages of a superscalar implementation without the expense of instruction scheduling hardware since all instructions are scheduled at compile-time. To an extent the cost and time-to-market advantages of programming in a high-level language and having

simple hardware outweigh the performance disadvantages of compiled code. Yet, improved performance is always desirable if achieved at a feasible cost.

Let us first take a brief look at the hardware utilisation by various DSP benchmarks of Texas Instruments' VLIW DSP TMS320C6201. Figure 1.1 shows the functional unit utilisation by some typical DSP benchmarks on the Texas Instrument's VLIW DSP TMS320C6201. The TMS320C6201 has two sets of 4 types of functional units - 40-bit ALU (L1 & L2), 16-bit multiplier (M1 & M2), 32-bit address generator and load/store units (D1 & D2) and 40-bit shifter (S1 & S2) [3]. The figure shows the proportion of time that a functional unit (e.g. L1) is busy, the proportion of time its complementary functional unit (e.g. L2) is busy as well as the proportion of time for which both these functional units (e.g. L1L2) are busy. In the ideal situation of all the functional units of the processor being fully utilised the statistics would be L1L2 = M1M2 = D1D2 = S1S2 = 100%. However, from Figure 1.1 we see that the utilization of all the functional units is extremely low. Figure 1.2 shows the statistics corresponding to the number of parallel or independent operations scheduled by the compiler every cycle. We see that the compiler, employing various software optimisation techniques like loop unrolling and software pipelining, does make an attempt at extracting parallelism and is at times able to schedule up to 6 parallel operations in a given cycle. However this does not happen too often and the functional units remain under-utilised most of the cycles. Therefore, in this work, we attempt to

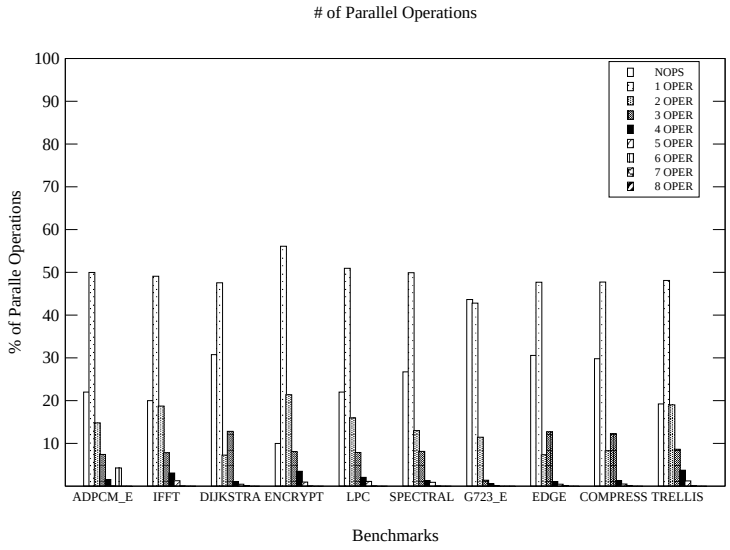


(a)

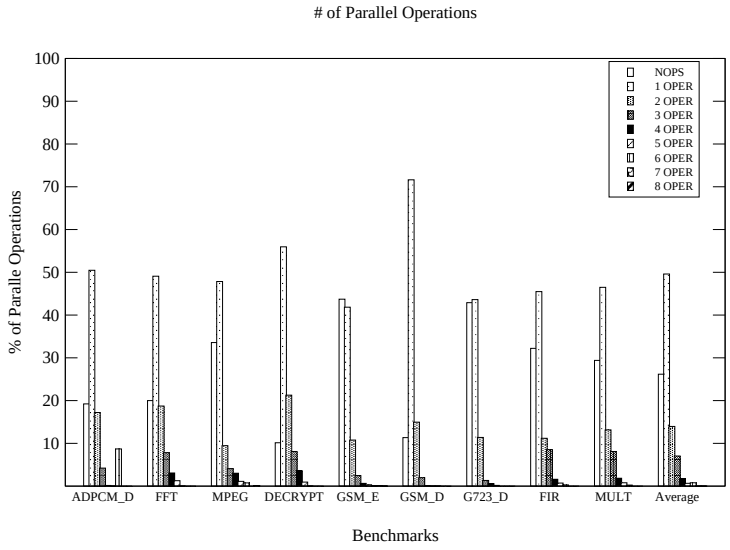


(b)

Figure 1.1: Functional Unit Utilisation of the VLIW DSP TMS320C6201



(a)



(b)

Figure 1.2: Number of Parallel Operations Scheduled for the TMS320C6201

improve the performance of a VLIW DSP by exploiting thread level parallelism (TLP) using simultaneous multithreading.

1.2 Simultaneous Multithreading

Simultaneous multithreading (SMT) tries to extract maximum parallelism by issuing as many instructions as possible from multiple threads in any given cycle [4] [5]. This helps to improve the processor throughput. These improvements can be seen in workloads that consist of mutually independent applications or applications that can be parallelized into independent threads by the programmer[6] [7] or the compiler. In addition, these performance improvements can be traded-off for lower power consumption by reducing the operating frequency and, therefore, voltage.

Incorporating SMT capability essentially involves replicating processor context (register files, program counter, etc.) for each thread but retaining the set of functional units and sharing them between the threads. Keckler and Dally have proposed an architecture called *Processor Coupling* where instructions from multiple VLIW threads are scheduled simultaneously to the functional units [8]. The compiler and architecture follow the assumption that the operations scheduled in a single VLIW instruction packet need not be executed

simultaneously. The hardware employs a scoreboard check mechanism to stall a given operation until all its source register operands are available. Thus, unlike a typical VLIW architecture, the hardware performs dependency check and resolves conflicts by stalling the processor. Later, Kaxiras, Narlikar et al have studied SMT in a VLIW DSP, wherein the hardware selects VLIW instruction packets from ready threads as many as can be accommodated, without splitting the VLIW instruction packets and assigns them to functional units [9]. We shall refer to their model as the Kaxiras model in later sections. Özer, Conte and Sharma also propose a SMT VLIW architecture named *Weld* [10]. The *Weld* architecture also selects VLIW instruction packets from ready threads and issues them to available functional units. The compiler embeds a separability bit in the opcode for each operation, which the hardware uses to decide if it can issue that operation separately splitting the VLIW instruction packet. The *Weld* architecture is aimed at increasing ILP using compiler-directed (speculative) threads in a multithreaded VLIW architecture.

In this work, we extend the idea of SMT VLIW to the next logical step, i.e. to be able to issue a subset of instructions from a VLIW instruction packet, based on the availability of functional units, without any explicit permission from the compiler. By doing this we would be able to schedule more operations to the execution units every cycle, thereby further improving the performance. However, a typical VLIW DSP compiler available commercially, like the TMS320C6000

compiler, schedules instructions with the assumption that all operations in a VLIW instruction packet will be issued for execution simultaneously. Therefore, by not issuing all the instructions in a VLIW packet in the same cycle, the VLIW compiler's data dependency assumptions could be potentially sabotaged. Hence, some mechanism would have to be provided to overcome this violation.

To solve this problem, we propose a limited dynamic scheduling technique using a set of delay-buffers to store results temporarily and commit them to the architectural register files at appropriate cycles. The dynamic scheduling technique proposed here is an extension of the *split-issue* technique proposed by Rau [11], which was originally intended to solve the backward compatibility problem for VLIW processors. The *split-issue* mechanism proposed by Rau essentially allocates a temporary buffer for an instruction to store its result and then commits it to the architectural register after N cycles, where (N+1) is the compiler-assumed-latency of the given instruction. This mechanism, therefore, permits correct execution of a program even when the actual latencies do not agree with the compiler-assumed-latencies. Rau's mechanism requires the compiler-assumed-latencies for a program to be conveyed to the hardware in some form.

Our mechanism is an extensions of Rau's. We make a key observation that in a VLIW architecture the latencies of operations are with respect to the

instruction packets and not machine cycles. We use this observation to extend Rau's *split-issue* mechanism to commit a result from the temporary buffer to the architectural register after N instruction packets, where (N+1) is the compiler-assumed-latency of the given instruction. That is, instead of counting the latencies in terms of cycles, we count latencies in terms of instruction packets. This allows the issue-hardware to split the instruction packets and issue individual operations within an instruction packet seperately or in any order. It is important to discern here that the hardware would still be unable to perform any out-of-order scheduling across different instruction packets. Also the issue-hardware would have to signal to the commit logic each time it finishes issuing all instructions in an instruction packet. Thus using this *extended split-issue* mechanism we would be able to issue subsets of instructions from instruction packets belonging to different threads based on functional unit availability.

1.3 Hardware-ISA Decoupling

Although it is not obvious at first, one of highlighting features of the hardware design proposed here is that it provides the flexibility to vary the number of hardware functional units of any given type. For example, the TMS320C6201 has 2 multipliers, M1 and M2, each connected to a different register file. For a wide

range of DSP benchmarks, it can be seen from Figure 1.1 that these multipliers are rarely utilised simultaneoulsy. Thus we could decide to have only one multiplier, with multiplexed connectivity to both register files, functioning as M1 or M2 as required. For instruction packets with two multiply operations, the issue-hardware would issue the second multiply in the following cycle. This, therefore, leads to the decoupling of the hardware implementation of a VLIW processor from its instruction set architecture (ISA). Backward compatibility is another important feature that can be directly derived from Rau's *split-issue* technique.

Results show that our *extended split-issue* mechanism expands the hardware design space in two dimensions for a VLIW processor. SMT capability coupled with the ability to have an optimal number of hardware functional units based on their utilisation presents novel opportunities to increase the effeciency of a VLIW DSP. Our experiments reveal that the proposed SMT VLIW DSP design provides significant processing gains. It is also shown that these could be traded-off for valuable reduction in energy consumption. Improved throughput efficiency is also seen by supplementing SMT capability with additional highly used functional units and decreasing the lesser used ones.

1.4 Organisation of the Thesis

The remainder of this thesis is organized as follows. In Chapter 2, we describe Rau's *split-issue* mechanism and our *extended split-issue* mechanism used to issue a subset of instructions from a VLIW instruction packet. In Chapter 3, we present the details of the proposed SMT VLIW architecture. In Chapter 4, we discuss the decoupling of the hardware implementation from the ISA and other beneficial features of the hardware design. Chapter 5 describes the simulation methodology and presents the performance results. Finally, our conclusions and scope for future work are presented in Chapter 6.

Chapter 2

Dynamic Scheduling in VLIW

2.1 Flow Dependences in VLIW Architecture

The original attraction of the VLIW architecture is its ability to exploit large amounts of ILP with relatively simple and inexpensive control hardware. It has been defined traditionally [11] by the following set of attributes.

- The ability to specify multiple independent operations in each instruction, to be issued for execution in the same cycle.
- The requirement for static, compile-time operation scheduling taking into account operation latencies and resource availability.
- Consequently, the requirement that the hardware conform exactly to the assumptions built into the program with regards to the number of functional units and operation latencies.
- The absence of any interlock hardware, despite the fact that multiple, pipelined operations are being issued every cycle.

Thus the VLIW architecture, when viewed as a contractual interface between the class of programs and the set of processor implementations, is basically an *Independence-Architecture* [12]. It specifies a set of operations that are guaranteed to be mutually independent and therefore to be issued simultaneously without any checks being made by the issue-hardware. We shall use the term execute packet (EP), as defined by Texas Instruments [13], to represent such an independent set of operations. These operations could have non-unit latencies. Therefore, the input operands for an operation are not determined by all the operations that were issued before the operation in question, but only those that have completed. In other words, earlier operations that have not yet been completed do not impose a flow dependence upon the given operation. Therefore, the compiler can and does reuse registers already assigned to long-latency operations for short latency operations during the intermediate cycles, to make more efficient use of registers. Consequently, any processor implementation would have to respect the flow dependences/independences in both directions: horizontal (within a single EP) and vertical (across different EPs).

In this chapter, we shall address the question of how to do limited dynamic scheduling in hardware without sabotaging the aforementioned flow dependences. The extent of dynamic scheduling would be limited to the ability to issue operations from a single EP over multiple clock cycles. However, in any given cycle, operations that might be issued simultaneously would always belong to the

same EP.

Cycle	EP	Operations
1	1	A5=add(A1,A2), A5=mul(A5,4), A5=load(A1)
2	2	A1=mul(A5,A5),
3	3	A6=sub(A1,A5), A7=load(A8)
4	4	...

Table 2.1: A VLIW Code Segment

Firstly, we shall look at the problems that would arise from issuing operations from a single EP over multiple cycles. Consider the given fragment of a VLIW program in Table 2.1, with operation latencies being 1 cycle for *add* and *sub*, 2 cycles for *mul* and 5 cycles for *load*.

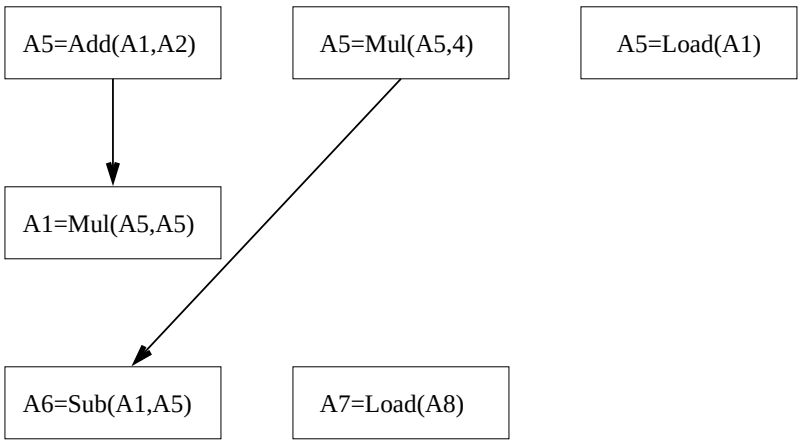


Figure 2.1: Dependence Flow for Register A5 For Code Segment in Table 2.1

In EP1, we see that all the operations write their results to the register A5. However, each of these operations would write their results to A5 in different cycles. The *mul* operation in EP1 should not get its input operand in register A5 from the *add* instruction in the same EP. Also, the *mul* operation in EP2 should get its input operand in register A5 from the *add* operation in EP1 and not from the *mul* or the *load* operations in EP1. Subsequently, the *sub* operation in EP3 should get its input operand in A5 from the *mul* in EP1 and not from the *add* or the *load* in EP1. Also, the input operand in A1 should not be from the *mul* in EP2.The data dependence for register A5 is shown in Figure 2.1.

Cycle	EP	Operations
1	1	A5=add(A1,A2)
2	1	A5=mul(A5,4)
3	1	A5=load(A1)
4	2	A1=mul(A5,A5)
5	3	A6=sub(A1,A5)
6	3	A7=load(A8)
7	4	...

Table 2.2: VLIW Code Segment With Each Operation Issued Seperately

However, if we issue each of the operations in a different cycle, as shown in Table 2.2, then the above flow dependences would not hold anymore. Figure 2.2

helps us understand that the *mul* in EP1 would get the output produced by the *add* operation in register A5. The *mul* in EP2 would get its input from the *mul* in EP1 instead of the *add* in EP1.

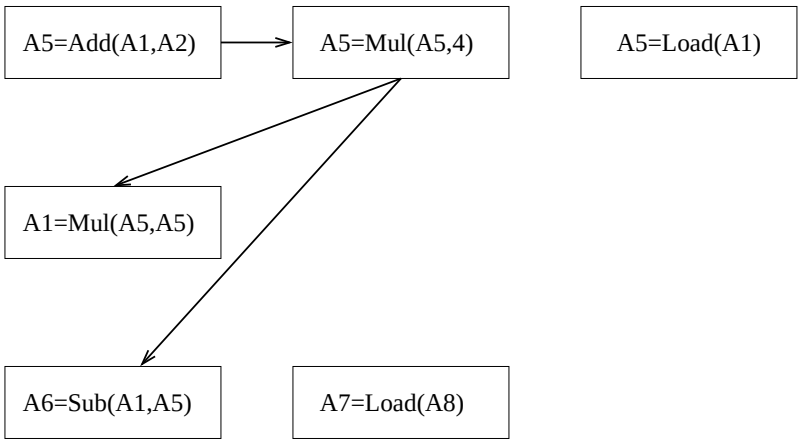


Figure 2.2: Incorrect Dependence Flow for Register A5 When Issuing Operations Seperately

2.2 Split-Issue Technique

To maintain the flow dependences we shall use an extended version of the *split-issue* scheduling technique proposed by Rau [11]. Rau's technique uses a set of buffers, called delay-buffers, to store results intermediately. These results are then committed to the architectural registers at appropriate cycles. This process is transparent to the programmer and the compiler. Rau's technique was originally proposed to solve the object code compatibility problem across VLIW

processors having different hardware latencies and, in fact, required that all operations in an EP be issued in the same cycle. We shall first illustrate the original process and then describe how it can be extended to handle the problems that would arise from issuing operations from a single EP over different cycles.

Cycle	EP	Operations	**Phase-2**
1	1	L1=add(A1,A2), M1=mul(A5,4), D1=load(A1)	**A5⇐L1**
2	2	M2=mul(A5,A5),	**A5⇐M1**
3	3	L2=sub(A1,A5), D2=load(A8)	**A6⇐L2, A1⇐M2**
4	4	...	
5	5	...	**A5⇐D1**

Table 2.3: VLIW Code Segment With Phase-2 Operations

In Table 2.3, we see a new set of operations, called phase-2 operations, augmenting the code fragment we saw earlier. The phase-2 operations specified along with each EP are carried out at the end of the given cycle. These essentially perform the job of committing data from the delay-buffers to the registers and are scheduled by the hardware to be carried out at the appropriate cycles. The outputs of all operations are stored in dynamically allocated delay buffers. In the above example the output of the *add* instruction in EP1 is stored in the delay-buffer L1. The phase-2 operation corresponding to this operation, i.e. committing the data in L1 to A5, is scheduled at the end of the same cycle.

Similarly, the output of the *mul* in EP1 is stored in delay-buffer M1 and then committed to A5 at the end of cycle 2. There are two phase-2 operations scheduled at the end of cycle 3 - the output of *sub* (from EP3) to be committed from L2 to A6 and the output of *mul* (from EP2) to be committed from M2 to A1. Thus, for every program operation, the corresponding phase-2 operations are scheduled after N cycles, where (N+1) is the latency of the given operation. We note that the program operations access the register file for their source operands but write their results in the allocated delay-buffers. Conversely, the phase-2 operations read from the delay buffers and copy the data into the appropriate architectural register.

2.3 Extended Split-Issue Technique

Now, we shall show that we could split the operations in an EP and issue them over different cycles without disturbing the flow dependences. To take care of this we issue the phase-2 operations only with respect to the last operation(s) issued from an EP, i.e. at EP boundaries. Hence, for every program operation, the corresponding phase-2 operations will be scheduled after N EP boundaries, where (N+1) is the latency of the given operation. This is because in VLIW architecture the flow dependences, horizontal and vertical, are really with respect to EPs and not clock cycles.

Cycle	EP	Operations	Phase-2
1	1	L1=add(A1,A2)	
2	1	M1=mul(A5,4)	
3	1	D1=load(A1)	A5⇐L1
4	2	M2=mul(A5,A5)	A5⇐M1
5	3	L2=sub(A1,A5)	
6	3	D2=load(A8)	A6⇐L2, A1⇐M2
7	4	...	
8	5	...	A5⇐D1

Table 2.4: VLIW Code Segment With Each Operation Issued Seperately Augmented By Phase 2 Operations

We can see from Table 2.4 that the *mul* operation in EP1 does not get its input operand in register A5 from the *add* operation belonging to the same EP but issued earlier. The mul operation in EP2 gets its input operand in register A5 from the *add* operation in EP1 (committed from L1) and not from the mul or the *load* operations in EP1. Moreover, the *sub* operation in EP3 gets its input operand in A5 from the mul in EP1 (committed from M1) and not from the *add* or the *load* in EP1. The input operand in A1 is not from the mul in EP2.

Thus, we see that by dynamically scheduling phase-2 operations for every

operation that is issued, at the appropriate EP boundary determined by the latency of the issued operation, we can maintain the flow dependences even if we do not issue all the operations in an EP in a given cycle.

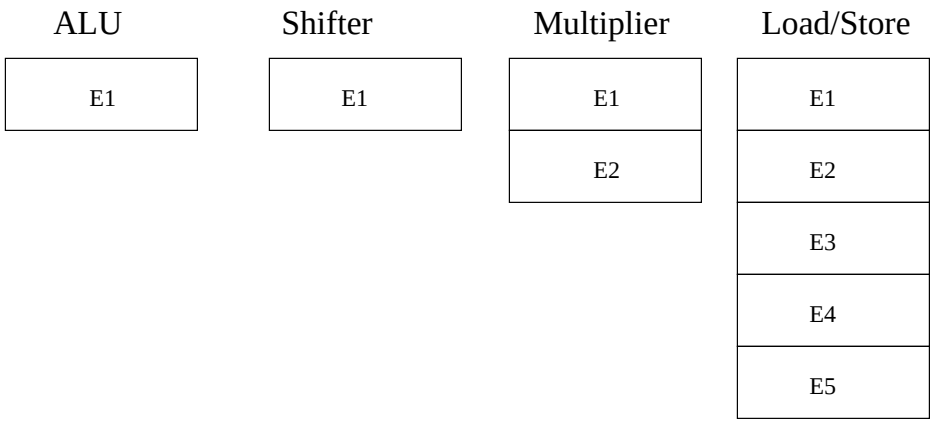


Figure 2.3: Execute Stages of the TMS320C6201

The next question that arises is how many delay-buffers would we require to be able to support the above technique. The number of instructions in the various stages of execution, in any given cycle, would determine the number of delay buffers required. Figure 2.3 shows the execute stages of the TMS320C6201 pipeline. For a given functional unit, the maximum latency of the operations handled, is the maximum number of instructions in its execute-pipeline. For example, the a multiplier has 2 stages in its execute pipeline, viz., E1 and E2. Therefore, in any given cycle there would be only 2 instructions being executed in the multiplier. This means that there would be atmost two outstanding phase-2 operations corresponding to the two instructions in the multiplier. The

number of functional units and associated latencies in the TMS320C6201 are given in Table 2.5.

Functional Unit	Description	Number of Units	Latency
L	ALU	2	1
M	Multiply	2	2
D	Load/Store	2	5
S	ALU/Shifter	2	1

Table 2.5: TMS320C6201 Functional Units and Their Latencies

Thus, the total number of delay-buffers required to support the *extended split-issue* mechanism on the TMS320C6201 are (2x1) + (2x1) + (2x2) + (2x5) = 18. To incorporate simultaneous multithreading, besides replicating the processor state (register file, program counter, etc.) for each thread, we would also need to replicate this set of 18 delay-buffers. We shall look into the architectural details of the simultaneous multithreaded VLIW processor in Chapter 3.

Chapter 3

Simultaneous Multithreaded VLIW Architecture

3.1 Base Architecture

In this chapter we shall first describe the base architecture chosen for this study, Texas Instruments TMS320C6201 [13][3], and then illustrate the modifications required to incorporate simultaneous multithreading using the *extended split-issue* technique detailed in the previous chapter.

The TMS320C6201 is an 8-wide fixed point VLIW DSP, with an 11-stage pipeline. It follows a clustered architecture, i.e. the 8 functional units are divided into 2 sets of 4 each. Two register files A and B, with 16 registers each, are connected to one set of functional units each. Crossovers allow limited use of the A-registers by the B-side functional units, and vice versa. Figure 3.1 depicts the CPU pipeline. The pipeline phases are divided into three stages, viz. fetch, decode and execute. The fetch phase comprises of 4 stages, program-address-generate (PG), program-address-send (PS),

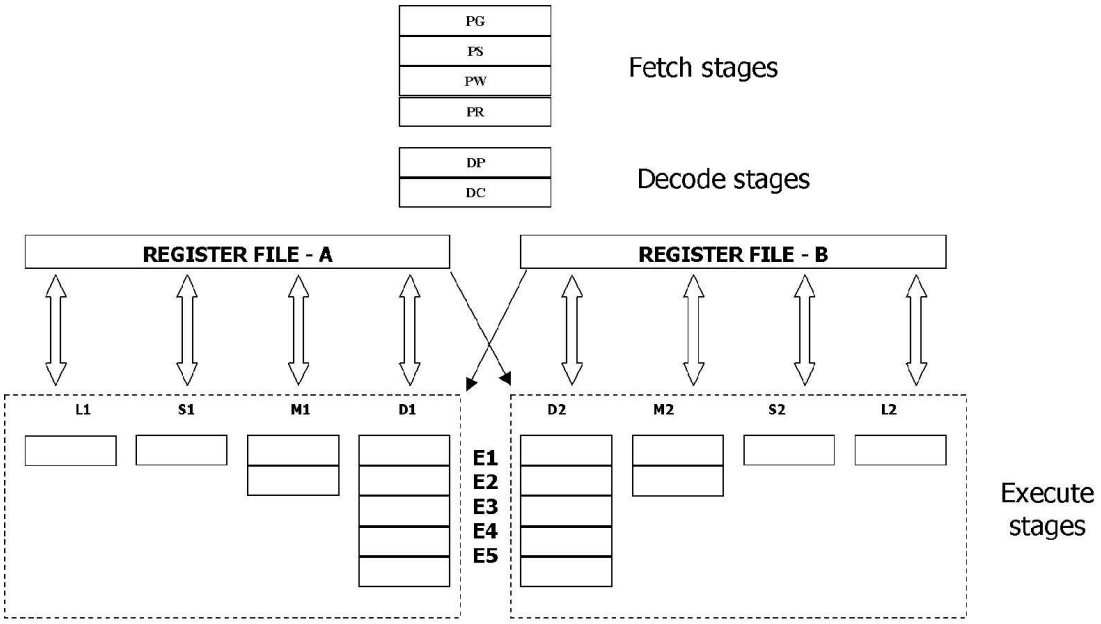


Figure 3.1: The VLIW Architecture (TMS320C6201)

program-access-ready-wait (PW) and program-fetch-packet-receive (PR). A fetch packet (FP) comprises of 8 operations (also referred to as instructions) packed together. However, all 8 operations need not constitute a single EP. An EP could be made up of any number of operations ranging from 1 to 8, and therefore a FP could comprise of a single EP or up to 8 EPs. The structure of a typical FP is shown in Figure 3.2.

The decode phase comprises of 2 stages, instruction dispatch (DP) and instruction decode (DC). During the DP stage, operations in an EP are extracted out of the FP and assigned to the appropriate functional units. At this point we note that if a FP contains more than one EP then the fetch stages stall until all

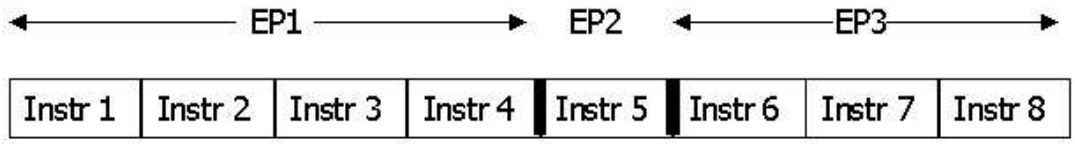


Figure 3.2: Execute Packets In A Fetch Packet

the EPs in that FP are dispatched. In the DC stage, the source registers, destination registers and associated paths are decoded for the execution of the operations in the functional units.

Finally, the execute phase involves a varying number of stages, depending on the functional unit. The 4 functional units on each side are a matched set. Each side contains a 40-bit integer ALU (L-unit), 40-bit shifter (S-unit), a 16-bit multiplier (M-unit) and a 32-bit adder also used in address generation for loads and stores (D-unit). As we can see from the figure, the L-units and S-units have an execution latency of 1 cycle, the M-units have a latency of 2 cycles and the D-units have a latency of 5 cycles. Branches are resolved in the S-units and take effect as delayed-branches with a delay of 5, i.e. the branch target begins execution (E1 stage) in the sixth cycle after the branch instruction. We can see from Figure 3.1 that, when the branch instruction is executed in the E1 stage of an S-unit, the target program counter (PC) is fedback to the PG stage. The branch can be said to be a single cycle latency operation, in effect, output of

which is not stored in any architectural register but in the program counter of the PG stage.

3.2 SMT Architecture

The SMT-VLIW architecture proposed by us is depicted in Figure 3.3. The figure shows an architecture that can support up to 4 threads. The processor context (i.e. register files, program counter etc.) is replicated appropriately to accommodate 4 threads. It resembles the base architecture in most aspects. It retains the same functional units, both in number and type. We see that the functional units get their input operands from the register files and store their output in the delay-buffers. The copyback-unit schedules the phase-2 operations to copy the data from the delay-buffers to the appropriate registers, upon receiving the *EP-boundary* signal. The DC stage generates the *EP-boundary* signal, when it decodes the last set of operations from an EP, for each thread.

Since the dynamic scheduling and multithreading happen in the decode stages we shall look at them first. To understand the decode stages, let us assume that the PR stage of the fetch phase provides fetch packets from all 4 threads. The DP stage is replicated so as to provide the ability to dispatch EPs from all 4 threads. The EP-combine stage, an additional stage depicted in Figure 3.3,

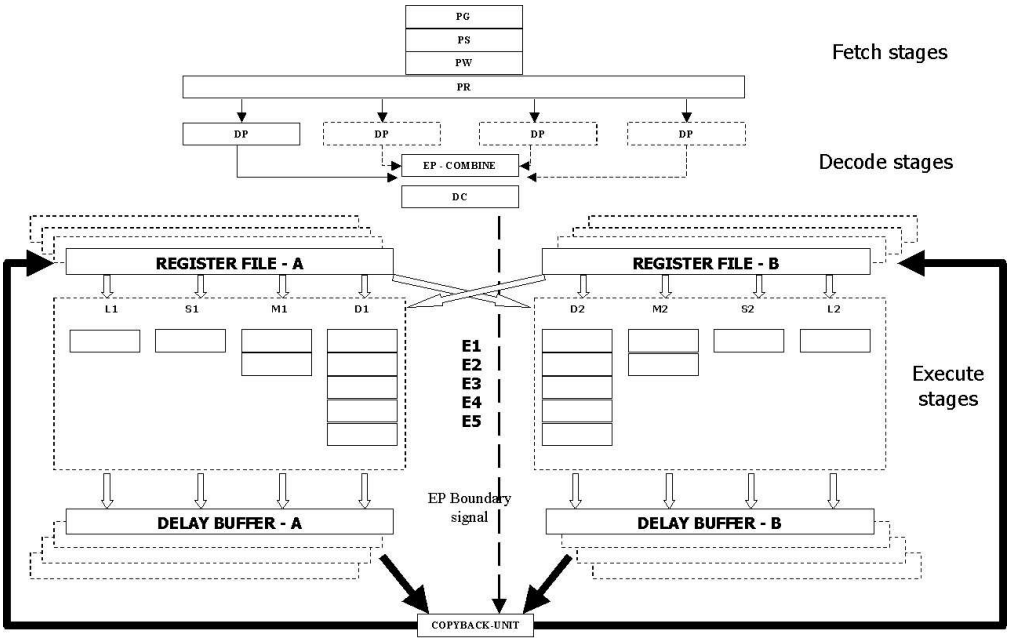


Figure 3.3: The Simultaneous Multithreaded VLIW Architecture

essentially represents the *extended split-issue* hardware that issues instructions from the EPs of different threads. The *extended split-issue* technique described in Section 2.3 is used to issue instructions from an EP for a given thread based on the availability of the functional units. The issue policy would determine the priority of the threads, and thereby the sequence in which they are to be chosen for issuing instructions. In this work, we use a fixed priority-based issue policy whereby thread 1 would be the highest priority thread, and all other threads would be secondary threads with decreasing priorities. Thus, the DC stage receives up to 8 instructions possibly from different threads to be decoded, each tagged with its thread identifier. For the instructions issued, the DC stage then decodes the source registers, destination registers and associated paths and sends

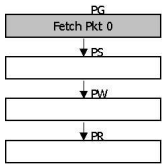
them to the different execution units. The source operands are read from the register files corresponding to the thread to which the instructions belong. When the DC stage decodes an instruction which is at an EP boundary it triggers an *EP-boundary* signal for that thread. The copyback unit keeps track of all the *EP-boundary* signals it receives and schedules the phase-2 operations for appropriate instructions based on their latencies in terms of EPs. Again, this is because in VLIW architecture the flow dependences are really with respect to the EPs and not clock cycles.

The functional units, after executing the instructions, store the results in the allocated delay-buffers. As all instructions are issued and executed in-order, the allocation of delay-buffers is simple and follows the in-order issue of instructions. Therefore, the phase-2 operations corresponding to the instructions are also scheduled in-order by the copyback unit.

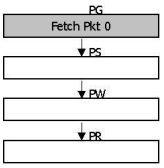
We note here that the SMT model proposed by Kaxiras et al [9] would also require a similar delay-buffer and copyback mechanism to maintain the vertical dataflow dependence assumptions of the compilers schedule. In the Kaxiras model, where the instructions are issued without splitting the EPs, the secondary threads may or may not be able to issue EPs every cycle based on the functional unit availability. This may, therefore, disturb the vertical data dependences for instructions in those threads, which would have to be taken care of with some

buffering mechanism. However, since the EPs are issued without splitting, the copyback logic would not require the DC stage to provide an *EP-boundary* signal and, therefore, would be simpler.

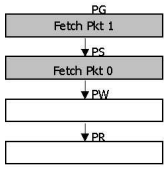
In the base architecture, we saw that the fetch phase stalls when a FP containing multiple EPs is being serviced by the DP stage. We exploit these stall cycles to fetch FPs from other threads. The clear advantage of doing this is that with minimal addition of logic we can fetch FPs from different threads without requiring additional fetch bandwidth. Figure 3.4, Figure 3.5 and Figure 3.6 elucidate the process of multiplexing the fetch phase between 2 threads. Figure 3.4 show the FPs 0, 1, 2 and 3 from thread 1 progressing through the fetch phases PG, PS, PW and PR, in cycles 0, 1, 2 and 3. We provide additional buffers in the PR stage to store multiple FPs corresponding to each thread as shown in Figure 3.4(h). In cycle 4, we see that the fetch phase in the base architecture stalls as the DP stage dispatches the first EP out of the FP in PR stage. Correspondingly, in the SMT architecture we use this cycle to fetch a FP from the next thread. Additional FPs coming into the PR stage are stored into the extra buffers. In cycle 5, Figure 3.5(c), as FP 0 has been completely dispatched, FP4 is fetched into the PG stage. The same happens in the SMT architecture as well. Again, in cycles 6 and 7, the fetch phase in the base architecture stalls as an EP is dispatched out of FP1. We utilize these cycles to fetch from the next thread. In cycle 7 we see that the FP0 for thread 2 has



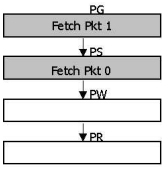
(a) Cycle 0 (Base)



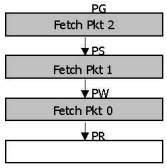
(b) Cycle 0 (SMT)



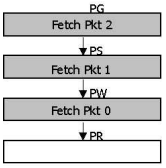
(c) Cycle 1 (Base)



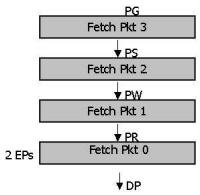
(d) Cycle 1 (SMT)



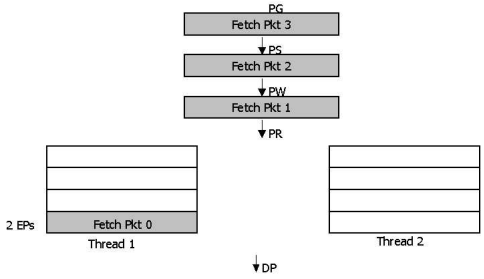
(e) Cycle 2 (Base)



(f) Cycle 2 (SMT)

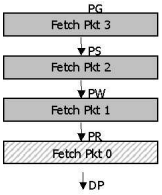


(g) Cycle 3 (Base)

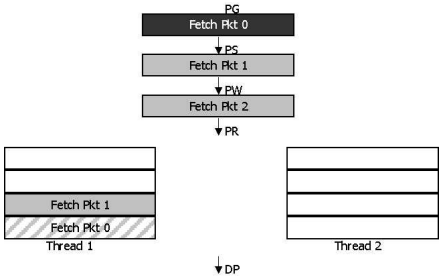


(h) Cycle 3 (SMT)

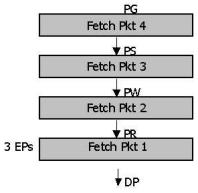
Figure 3.4: Comparison of fetch stages in Base Architecture and the SMT Architecture



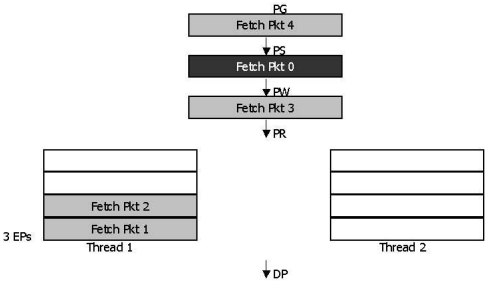
(a) Cycle 4 (Base)(As an EP is dispatched out of FP0 - Fetch stages stalled)



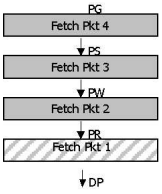
(b) Cycle 4 (SMT)(As an EP is dispatched out of FP0 - Fetch stages multiplexed)



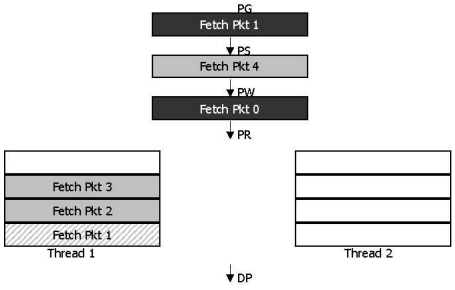
(c) Cycle 5 (Base)



(d) Cycle 5 (SMT)

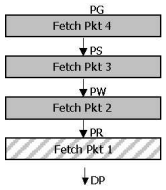


(e) Cycle 6 (Base)(As an EP is dispatched out of FP1 - Fetch stages stalled)

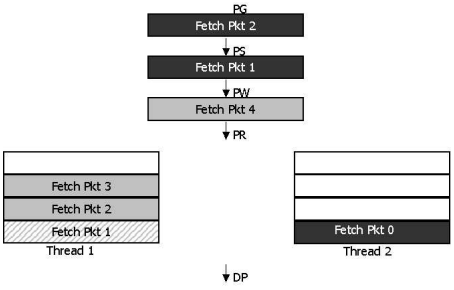


(f) Cycle 6 (SMT)(As an EP is dispatched out of FP1 - Fetch stages multiplexed)

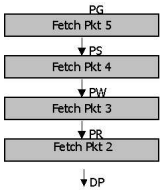
Figure 3.5: Comparison of fetch stages in Base Architecture and the SMT Architecture (contd.)



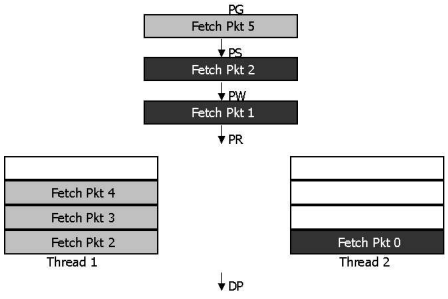
(a) Cycle 7 (Base)(As an EP is dispatched out of FP1 - Fetch stages stalled)



(b) Cycle 7 (SMT)(As an EP is dispatched out of FP1 - Fetch stages multiplexed)



(c) Cycle 8 (Base)



(d) Cycle 8 (SMT)

Figure 3.6: Comparison of fetch stages in Base Architecture and the SMT Architecture (contd.)

reached the PR stage. However, the DC stage would not automatically start dispatching EPs from the FPs in the PR stage. The DC stage would dispatch EPs from a given thread only when there are atleast 4 FPs of the given thread in the fetch stage buffers. This is required to maintain the delayed-branch latency.

In the PR stage, the FPs are stored in separate sets of buffers corresponding to the thread number. We could extend this process to accommodate any number of threads. It is to be noted that the highest priority thread would not fetch when there are multiple EPs in the FP being dispatched by the DP stage, just like it happens in the base architecture. Thus, the simultaneous multithreading happens transparently with respect to the highest priority thread. However, for the lower priority threads this is not the only condition when they cannot fetch a FP. A low priority thread would be unable to fetch and dispatch in the DP stage when the EP-combine stage would be unable to issue a complete EP from that thread. Ofcourse, in a given cycle a low-priority thread would be unable to fetch if a higher priority thread is ready to fetch.

Thus design of the SMT VLIW processor ensures that for any given thread all threads with lower priority are transparent. Hence the instructions of the thread with the highest priority would flow through the SMT VLIW processor oblivious of instructions from other threads and be executed as if it is being executed in the base processor, TMS320C6201.

Chapter 4

Hardware-ISA Decoupling

Besides the ability to incorporate SMT, a significant highlighting feature of the *extended split-issue* is that it provides the leverage to decouple the hardware implementation of the VLIW processor from its instruction set architecture (ISA). That is, the compiler could schedule instructions based on the ISA, but the hardware implementation need not conform exactly to the architectural structure. This is an extremely unorthodox concept for VLIW processors, and it presents an enormous potential for future designs. We have noted earlier that the dataflow dependences in the VLIW architecture are with respect to the EPs and not clock cycles. We have shown that since the delay-buffer and copyback logic schedule phase-2 operations to commit the results of instructions based only on the *EP-boundary* signal, the issue logic can afford to break the EPs and still not disturb the compiler's semantics.

To understand how this gives us the freedom to decouple the hardware implementation, let us again look at the TMS320C6201. Again, Figure 1.1 shows

the typical functional unit utilisation statistics for the various DSP benchmarks. We observe that the processor has 2 multipliers, M1 and M2, and both have very low utilisation. Moreover, both the multipliers are very rarely used simultaneously. However, they cannot be completely removed as DSP applications require fast multiplications, which in this case take just 2 cycles. But having two multipliers seems to be an overkill, especially as these fast multipliers have complex and large amounts of logic consuming a considerable amount of die area and power. Therefore, we could design an implementation of the TMS320C6201 with only one multiplier. In any given cycle, this multiplier could function as M1 or M2 as required. However, if any EP has instruction scheduled for both the multipliers, then the issue logic would split the EP and issue one of them the following cycle. Thus, the program would behave as if one of the multipliers is always being used by a (imaginary) higher priority thread. However, such a design requires that this multiplier has datapaths multiplexed to both the register files allowing it to behave like M1 or M2 as required in a given cycle.

This ability is a feature of the *extended split-issue* mechanism requiring delay-buffers and copyback logic. To understand the cost of implementing the *extended split-issue* mechanism, we look at Kaxiras' model of SMT VLIW. Kaxiras' model does not support hardware-ISA decoupling. We noted in the previous chapter that the Kaxiras model also would have to employ as many delay-buffers, albeit with simpler copyback logic. Thus, we can see that with a

small increase in complexity of the copyback logic, we gain the twin abilities of SMT and hardware-ISA decoupling.

4.1 Salient Features of the SMT VLIW

The SMT architecture has other salient features worth mentioning, viz. backward compatibility and ability to hide cache latencies. Although we haven't provided for these features in our SMT design, each of these can be incorporated with minimal amount of additional logic. Let us take a brief look at these.

Since catering to object code compatibility across processors with different hardware latencies was the original motivation for Rau's *split-issue* technique [11], it can be easily derived from our design too. The latency assumed for each operation, by the compiler, could be specified to the hardware in a number of ways. These could be listed, in decreasing order of generality and flexibility, as follows.

- a field in each operation specifying the assumed latency,
- an execution latency register (ELR) per opcode or per functional unit
- an architecturally specified latency for each opcode

The first approach is too extravagant in its use of instruction bits and the last

approach too rigid, making it impossible to accomodate latency changes within an architectural family. The second approach seems to be well balanced and can be adopted with minimal cost. It would require an ELR for each opcode or functional unit, which could be loaded by hardware from the header of the program. A more dynamic option could be to make the ELRs visible to the program and provide opcodes that load the latencies in them. Thus, the copyback unit would use the ELR values to schedule the phase-2 operations appropriately. If the assumed latencies happen to be smaller than the actual latencies then stalling the issue of instructions for the appropriate number of cycles would be the simplest way to ensure that the dataflow dependences are maintained correctly.

As a direct benefit from multithreading, TLP can be exploited to utilise the processor functional units during cache miss latencies. Although traditional DSP systems did not employ caches, there have been recent studies on the use of caches [14]. This is because the novel trend of programming DSPs in high-level languages has exposed the traditional non-uniform DSP memory model as a weakness, as it makes for a poor compiler target [15]. However, caches introduce unpredictable latencies in memory operations. Also, the additional delay in memory access due to cache-misses would require the entire pipeline to stall, to avoid any dependence conflicts, thus wasting those idle cycles. The SMT design proposed here can be used to overcome this very easily. When a memory access

in a particular thread encounters a cache-miss, the functional units can still continue their execution and write their results in the delay-buffers. The thread waiting on a cache miss would be stalled, i.e. not be allowed to fetch any new FPs or issue fetched ones to the functional units. Meanwhile, other threads would continue their execution in regular fashion, having one thread less to compete with. The phase-2 operations for the waiting thread would stall till the memory access is completed. This would ensure that there would be no dataflow dependence conflicts. The arriving data can be directed to the appropriate delay-buffer and the thread can continue its normal flow. Thus, TLP is exploited to overcome throughput inefficiencies due to cache-miss latencies.

Chapter 5

Simulations and Results

5.1 Simulation Methodology

In this section, we describe the DSP benchmarks used for evaluating our design. We also present the experimental setup used.

Our choice of DSP benchmarks spans three benchmark suites, viz., MediaBench [16], MiBench [17] and the UTDSP benchmark suite [18]. We have chosen a wide range of benchmarks. These benchmarks do encompass the general behaviour of DSP algorithms for the different simulation studies carried out. We shall describe each of these benchmarks in brief.

MediaBench:

MPEG: Comprises of MPEG2 decoder. The important computing kernel is an inverse discrete cosine transform for decoding. GSM: European GSM 06.10 provisional standard for full rate speech transcoding, prI-ETS 300 036. It compresses frames of 160 13-bit samples (8KHz sampling rate, i.e., a frame rate

of 50Hz) into 260 bits. G723: Reference implementation of the CCITT G.723 voice compressions. PEGWIT: Comprises of encrypt and decrypt, programs for public key encryption and authentication. It uses an elliptic curve over GF(2^{255}), SHA1 for hashing, and the symmetric block cipher square.

MiBench:

FFT/IFFT: Performs a Fast Fourier Transform and its inverse. The input data is a polynomial function with pseudorandom amplitude and frequency components. ADPCM encode/decode: Adaptive Differential Pulse Code Modulation is a variation of the well-known standard Pulse Code Modulation. It takes 16-bit linear PCM samples and converts them to 4-bit samples, yielding a compression rate of 4:1. The input data are small and large speech samples. Dijkstra: This benchmark constructs a large graph in an adjacency matrix representation and then calculates the shortest path between every pair of nodes using repeated application of Dijkstra' algorithm.

UTDSP:

LPC: The program implements a Linear Predictive Encoder. TRELLIS: The program implements a Trellis decoder. SPECTRAL: This application calculates the power spectral estimate of speech using periodograms-averaging. EDGE DETECT: This benchmark detects the edges in a gray-level 128x128 pixel image. The program relies on a 2D-convolution routine to convolve the image with Sobel operators that expose horizontal and vertical edge information. COMPRESS: This program uses the discrete cosine transform to compress a 128x128 pixel

image by a factor of 4:1 while preserving the information content. FIR: Implements a 256 tap FIR filter. MULT: Implements matrix multiplication over two 10x10 matrices.

The C implementations of the benchmarks were compiled by TI's commercial TMS320C6000 compiler [19]. Significant compiler optimisations that were activated, by using the flag *-o2*, can be listed as follows:

- Software pipelining
- Loop unrolling
- Eliminating global and local common subexpressions
- Eliminating global and local unused assignments
- Converting array references in loops to incremented pointer form
- Performing loop rotation
- Performing loop optimisations
- Performing control-flow-graph simplification
- Allocating variables to registers

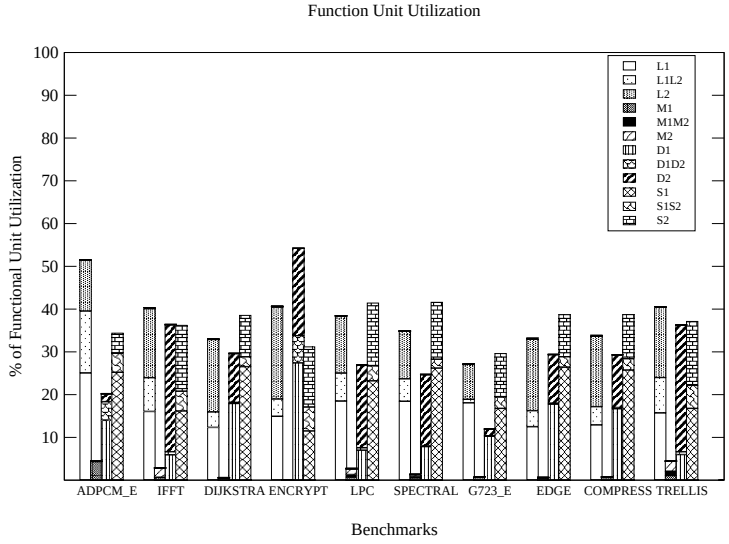
We conducted the simulations on a cycle-accurate, instruction level simulator of the TMS320C6201 [20]. We extended the original simulator to incorporate

SMT capability as described in Chapter 3. The benchmarks were executed for 50M cycles at 200MHz representing a workload of 0.25sec. The operating voltage at this frequency was taken to be 2.5V [13]. To study the energy consumption in the SMT model operating at 200MHz, we maintain the workload for each benchmark for 2, 3 and 4 threads, by executing them for appropriately lesser number of cycles as the throughput increased. In investigating the energy consumption at lower frequencies, the operating frequencies were scaled down to 125MHz, 105MHz and 95MHz for 2, 3 and 4 threads respectively. These frequencies reflect the average gain in IPC in 2, 3 and 4 threads. Again, each of the benchmarks was executed for an appropriate number of cycles to maintain the workload. The reduction in operating voltages [21] was calculated using the relationship: time-delay $\propto V_{dd}/(V_{dd} - V_t)^{\beta}$. We have taken $\beta = 1.4$, as a reasonable assumption. For operating frequencies of 125MHz, 105MHz and 95MHz, for $V_t = 0.7V$, the operating voltages (V_{dd}) were calculated to be 1.65V, 1.49V and 1.40V. As prescribed in the Wattch framework [22], the activity factors for different sub-blocks of the processor were determined every cycle and used in the calculation of dynamic power. The energy consumption is calculated as the summation of this dynamic power consumed over the entire workload duration.

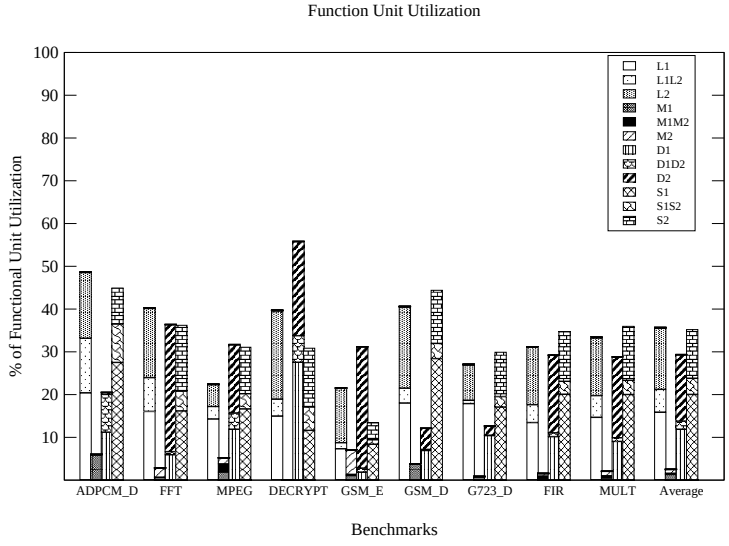
5.2 Results

First, let us look at the utilisation statistics of the TMS320C6201. For a wide variety of DSP benchmarks, Figure 5.1 shows the proportion of time that one functional unit is busy (eg. L1), the proportion of time its complementary functional unit is busy (eg. L2) and the proportion of time during which both functional units are simultaneously busy (eg. L1L2). We observe that the utilisation of the functional units is extremely low. Figure 5.2 shows the statistics corresponding to the number of parallel or independent operations scheduled by the compiler every cycle. It shows the proportion of time for which no operations (nops) are scheduled, proportion of time only 1 operation is scheduled, proportion of time only 2 operations are being scheduled and so on. These statistics show that the compiler does make an attempt at extracting parallelism and is at times able to schedule up to 6 parallel operations in a given cycle. However, this does not happen too often and the functional units remain under-utilised most of the cycles.

Figure 5.3 shows the average functional unit utilisation for all the above mentioned benchmarks for increasing number of threads. We see that as we increase the number of threads to 4, the utilisation of the L, S and D units increases significantly. Also, the proportion of time for which the pairs of L, S and D units are simultaneously utilised increases. Figure 5.4, shows the proportion of

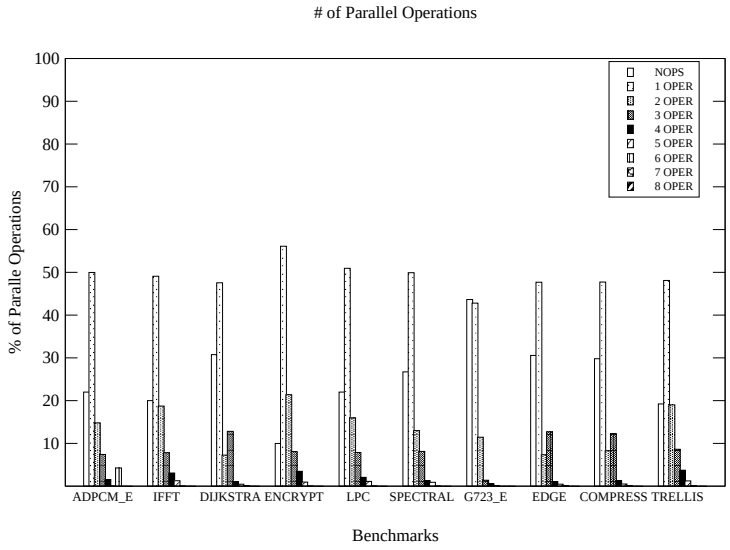


(a)

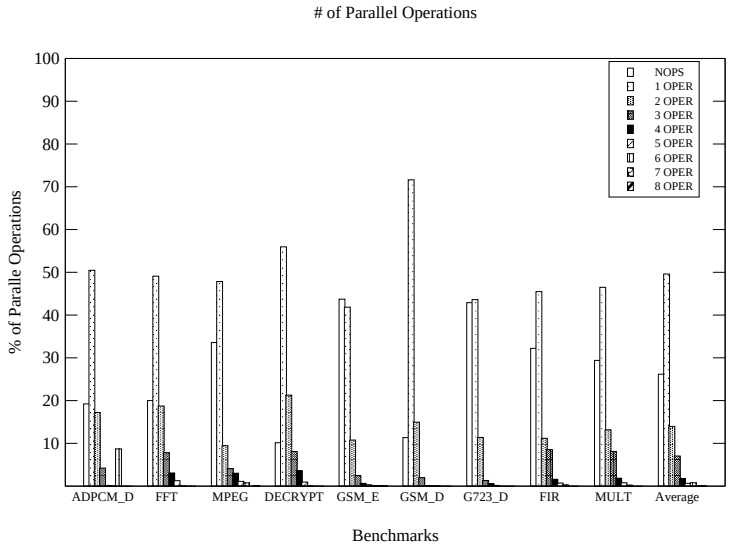


(b)

Figure 5.1: Functional Unit Utilisation of the VLIW DSP TMS320C6201



(a)



(b)

Figure 5.2: Number of Parallel Operations Scheduled for the TMS320C6201

time for the number of operations issued for execution every cycle, on an average. For a single thread, we see that for nearly 90% of the cycles less than 2 operations are issued every cycle. As the number of threads are increased to 4, it can be seen that for nearly 75% of the cycles at least 2 operations are issued every cycle.

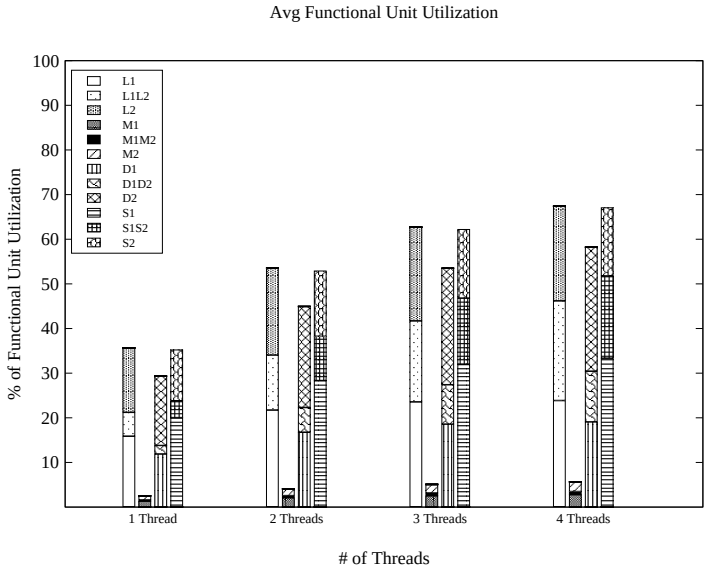


Figure 5.3: Average Functional Unit Utilisation for the SMT Architecture for Various Threads

In Figure 5.5, we observe that despite the numerous compiler optimisations, IPC for the base architecture is just over 1.0 on an average. For an 8-wide VLIW, this indicates a very low processing throughput. One of the reasons for this could be attributed to the high number of NOPs required to pad the unused slots in a FP, as can be seen from Figure 5.2. It is seen that, on an average, nearly 25% of cycles are consumed by NOPs. Another factor that contributes to the low IPC in

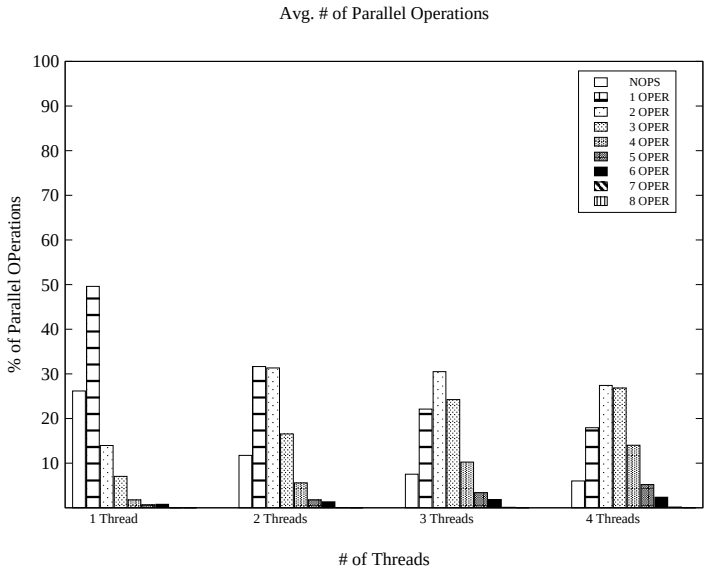


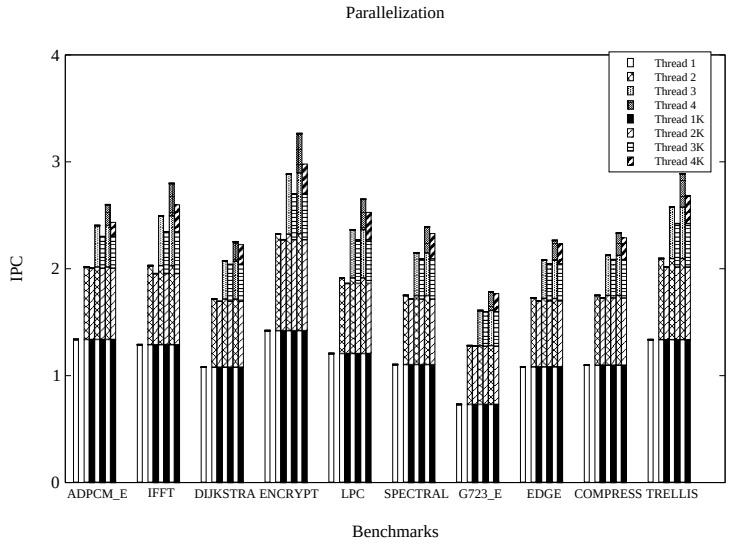
Figure 5.4: Average Number of Parallel Operations for the SMT Architecture for Various Threads

the TMS320C201 is that each of the 2 register files have only 16 registers. TI, in its next generation DSP TMS320C64xx, has shown that doubling the register file sizes could contribute to significant increases in the processing throughput [23]. The following is a small code segment extracted from the benchmark Decrypt (PEGWIT) compiled by the TMS320C6201 compiler. We see here in this example that most of the EPs are quite short. We observed that this is the case in all the benchmarks. The detailed code listing for 3 benchmarks Decrypt, GSM_Encode and MPEG_Decode are provided in the Appendix.

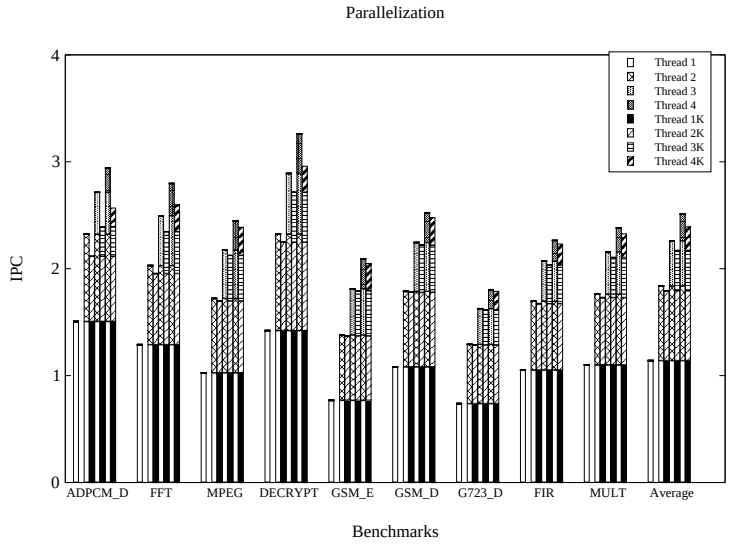
```
_gfInvert:
;** --------------------------------------------------------------------------*
           LDW     .D2T2   *+DP(_logt),B0    ; |388|
           ADDK    .S2     -248,SP           ; |381|
           NOP             2
```

```
           STW     .D2T2   B3,*+SP(232)      ; |381|

   [ B0]   B       .S1     L80               ; |388|
||         STW     .D2T1   A11,*+SP(228)     ; |381|

           STW     .D2T1   A10,*+SP(224)     ; |381|
           STW     .D2T2   B12,*+SP(244)     ; |381|
           STW     .D2T2   B11,*+SP(240)     ; |381|

           STW     .D2T2   B10,*+SP(236)     ; |381|
||         MVKL    .S2     RL80,B3           ; |388|

           MVKH    .S2     RL80,B3           ; |388|
||         MV      .L1X    B4,A11            ;
||         MV      .S1     A4,A10            ;
||         STW     .D2T2   B13,*+SP(248)     ; |381|

           ; BRANCH OCCURS                   ; |388|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |388|
           MVKL    .S1     SL22+0,A4         ; |388|
           MVKH    .S1     SL22+0,A4         ; |388|
           NOP             3
RL80:      ; CALL OCCURS                     ; |388|
;** --------------------------------------------------------------------------*
L80:
           LDW     .D2T2   *+DP(_expt),B0    ; |388|
           NOP             4
   [ B0]   B       .S1     L81               ; |388|
   [ B0]   MV      .L1     A10,A1
           NOP             4
           ; BRANCH OCCURS                   ; |388|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |388|
           MVKL    .S1     SL22+0,A4         ; |388|
           MVKL    .S2     RL82,B3           ; |388|
           MVKH    .S1     SL22+0,A4         ; |388|
           MVKH    .S2     RL82,B3           ; |388|
           NOP             1
RL82:      ; CALL OCCURS                     ; |388|
           MV      .L1     A10,A1
;** --------------------------------------------------------------------------*
```

Figure 5.5 shows the processor throughput (IPC) of our model of SMT VLIW for 2, 3 and 4 threads in comparison with the Kaxiras model where the instructions are issued without splitting the instruction packets. In the figure,



(a)



(b)

Figure 5.5: Processor Throughput (IPC) of the SMT VLIW

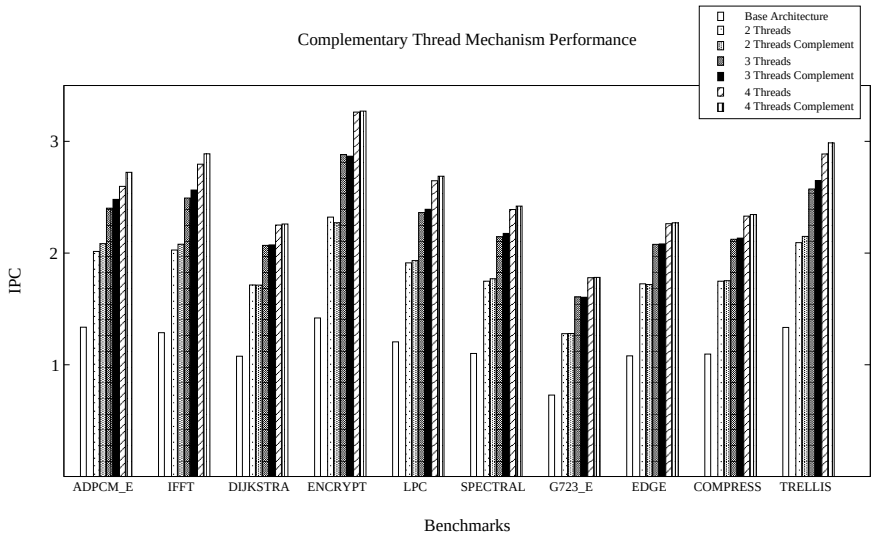
each histogram shows the additional gain in IPC, in a staggered manner, for every additional thread supported. We observe that our model performs at least as well as the Kaxiras model and on an average 2.2%, 4% and 4.8% better for 2, 3 and 4 threads. Compared to the base architecture, both the SMT configurations show substantial gains in IPC. The average gains in IPC for our SMT model are about 61%, 98% and 120% for 2, 3 and 4 threads, respectively.

Even though we do not see a significant difference in the processing throughput of our SMT model in comparison with the Kaxiras model, it is the ability to vary the number of hardware functional units that distinguishes our design. As we shall see ahead, it is this ability, coupled with SMT, that holds enormous potential for future designs.

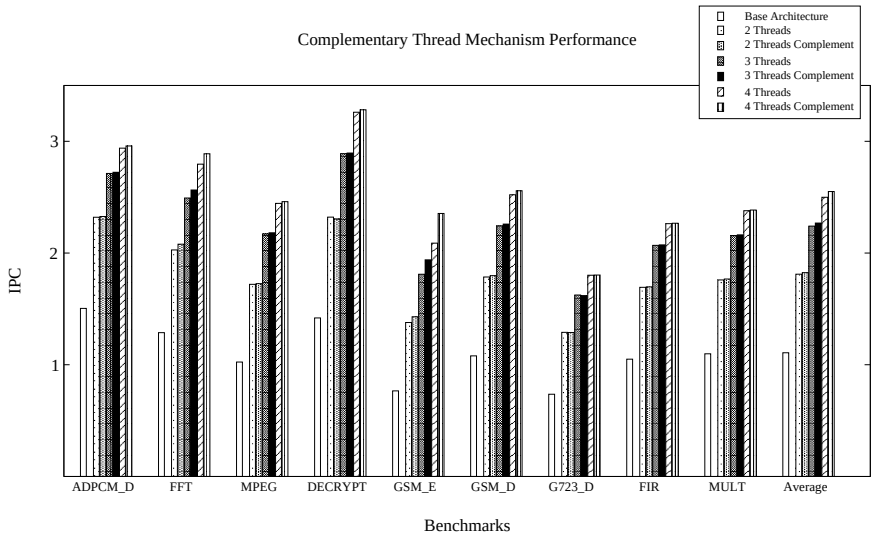
Also, we investigated any bias in the compiler's schedule towards a particular set of functional units i.e. A side or B side. We studied the behaviour of the SMT processor by alternating the issue of instructions to the A-side or B-side functional units based on the thread number. Instructions in thread 2 and 4 would be issued in opposite fashion, i.e. A-side instructions would be issued to B-side units and B-side instructions would be issued to A-side units. We see from Figure 5.6 that there is no improvement or difference in the processing throughput of the SMT configured to issue instructions in such a complementary fashion for alternate threads against the SMT with natural configuration.

Next, in Figure 5.7, we see the behaviour of the benchmarks over different hardware configurations, viz., removing a multiplier (-M), removing a multiplier and adding an ALU (-M+L) and removing a multiplier and adding an ALU and Shifter (-M+L+S). We chose to add additional L and S units as we observed from Figure 5.3 that the L and S units were highly utilised, especially as the number of threads was increased. We see that for a single thread the decrease in performance by removing a multiplier is insignificant. However, for increased number of threads, removing a multiplier does impact the performance. Adding an ALU in place of a multiplier (-M+L) compensates well and boosts up the throughput. Further gains in throughput are seen in the -M+L+S hardware configuration. An interesting observation, here, is that for a couple of benchmarks, ADPCM_decode and GSM_Decode, the -M+L+S configuration produces greater throughput for 3 threads as compared to executing 4 threads in the unchanged configuration of the processor. This observation brings us the interesting possibility that an SMT VLIW with support for lesser number of threads with appropriate number of additional functional units might be a *better* design choice over an SMT VLIW with support for more number of threads but retaining the original number of functional units.

We, also, carried out some experiments with real-time behaviour of the DSP workloads. Figure 5.8 shows the behaviour of 3 DSP benchmarks, viz.

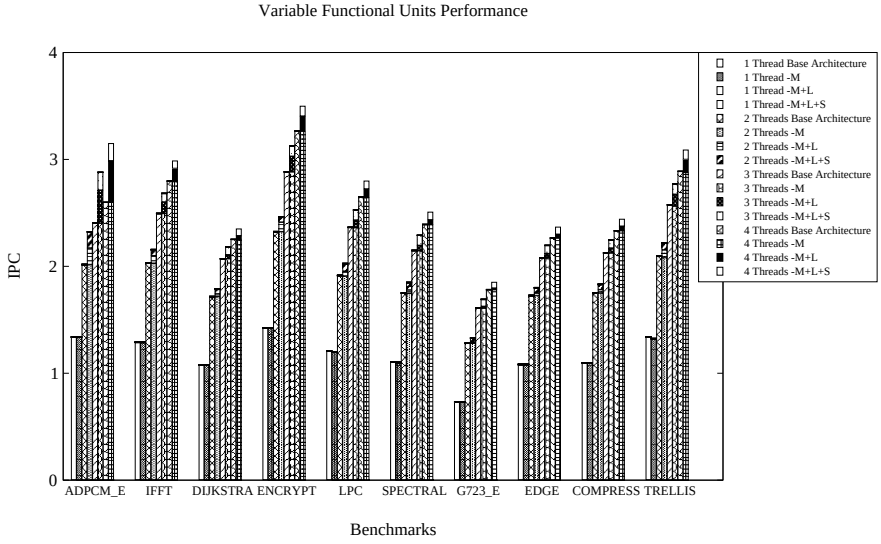


(a)

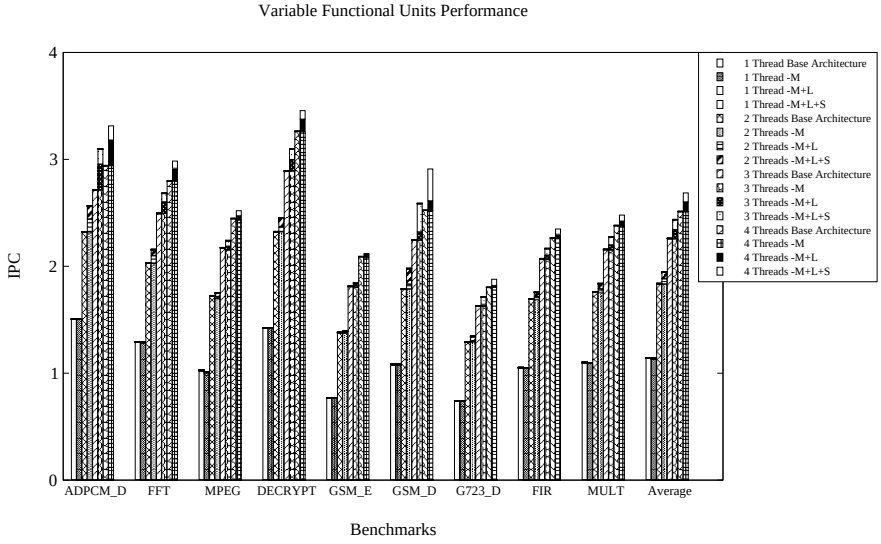


(b)

Figure 5.6: IPC of the SMT VLIW for Complementary Issue



(a)



(b)

Figure 5.7: IPC of the SMT VLIW for Different Hardware Configurations

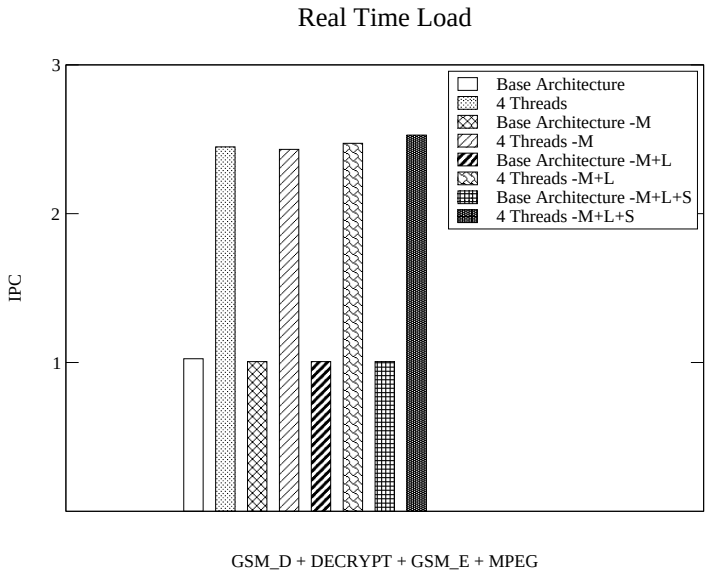
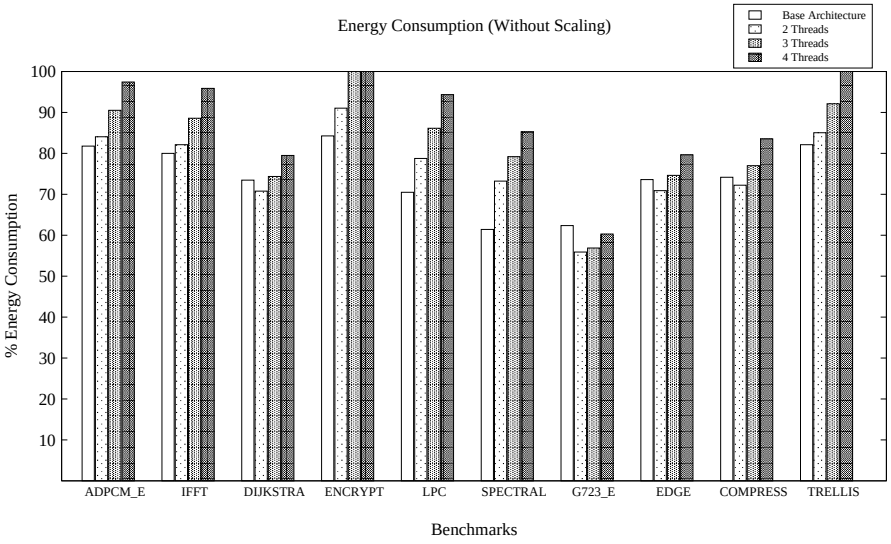
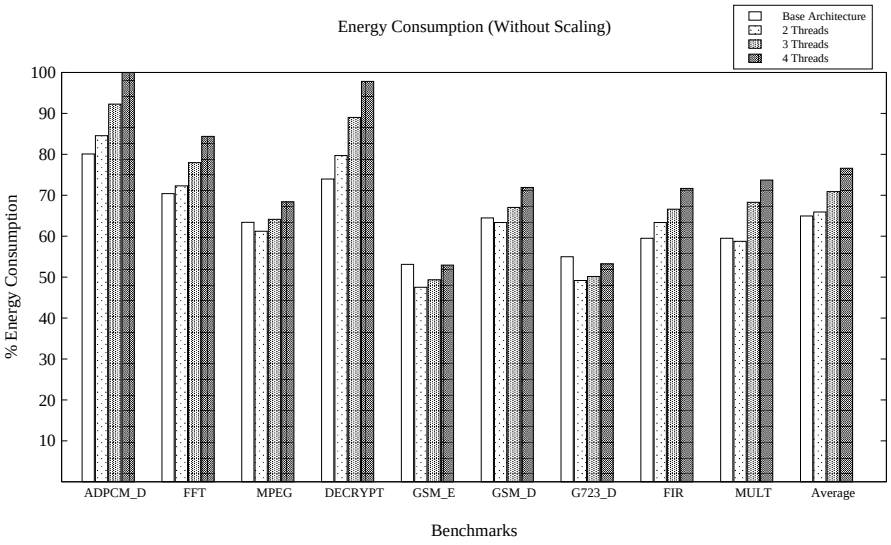


Figure 5.8: Real Time throughputs for a combination of DSP benchmarks

GSM_Encoder, Decrypt and GSM_Decoder, working at the rate of 50 frames per second and the MPEG decoder at 3 frames per second. In the base architecture, for every period of 20ms (4M cycles), the workload is arranged in such a manner that the benchmarks are executed in a serial fashion in the sequence - GSM_Encoder, Decrypt, GSM_Decoder and the MPEG decoder consumes the remaining available cycles. For the SMT version, again for every period of 20ms (4M cycles), the GSM_Encoder, Decrypt, GSM_Decoder and the MPEG decoder each execute as independent threads. Also, after the first 3 benchmarks finish their execution, the MPEG decoder is executed for the remaining number of cycles in each of those threads as well. We can see from Figure 5.8 that there is a significant increase in the processing throughput in the SMT version (4 threads)

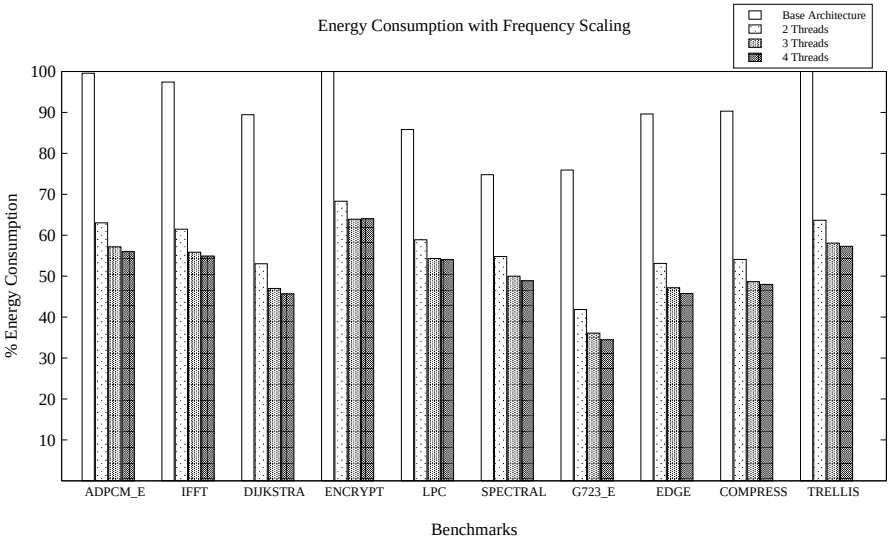


(a)

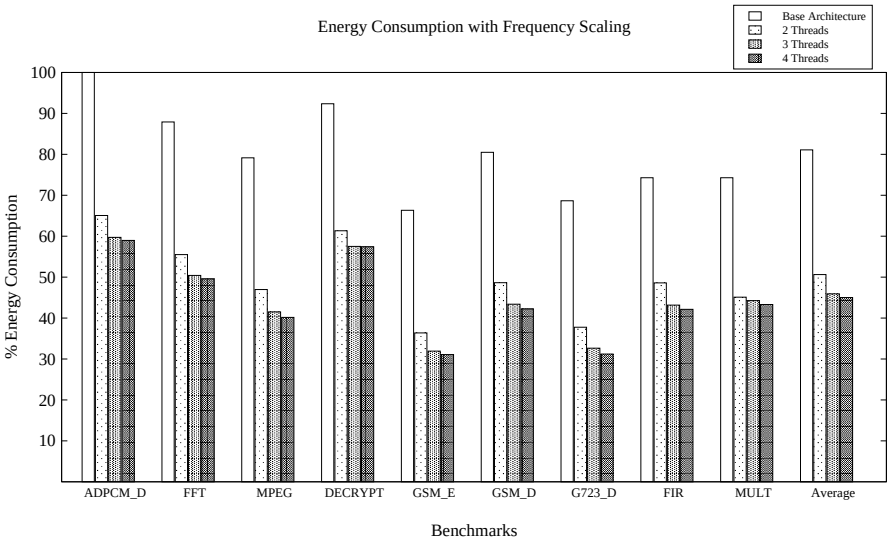


(b)

Figure 5.9: Energy Consumption for 1, 2, 3 and 4 Threads

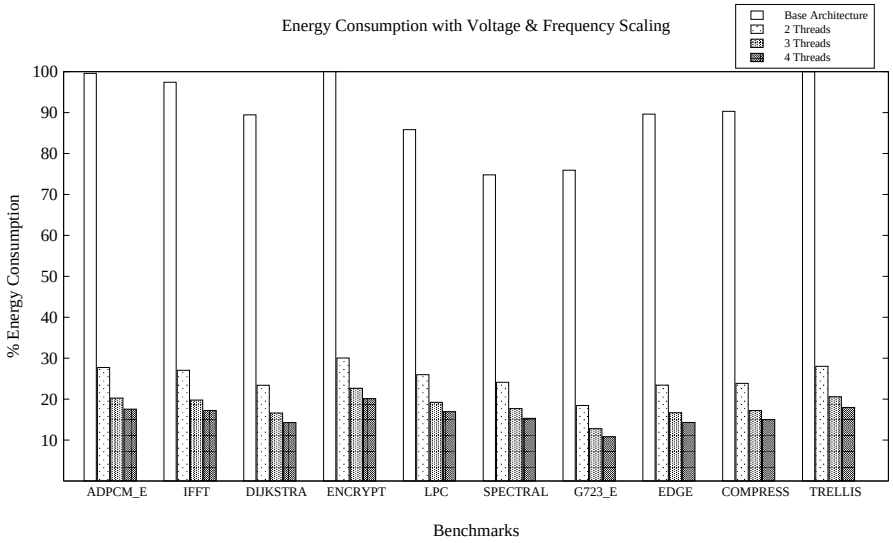


(a)

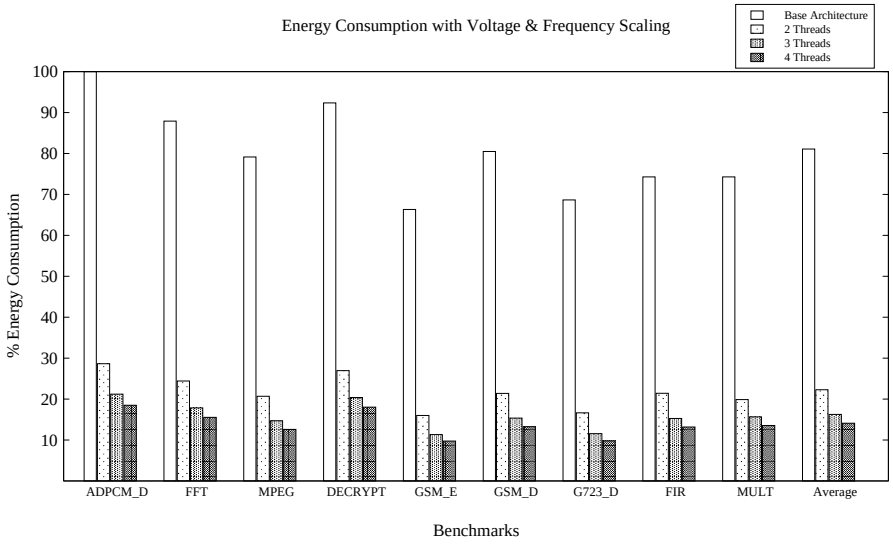


(b)

Figure 5.10: Frequency Scaled Energy Consumption for 1, 2, 3 and 4 Threads



(a)



(b)

Figure 5.11: Frequency and Voltage Scaled Energy Consumption for 1, 2, 3 and 4 Threads

as compared to the base processor. Also, we ran these simulations for different configurations of the hardware. These results show that the increased processing throughput gained in our various design configurations of the SMT VLIW hold their promise for workloads comprising of a combination of DSP applications as well.

Figure 5.9 shows the energy consumption of a given workload executed as 1, 2, 3 or 4 threads. Quite naturally, we see that as we increase the number of threads the energy consumed is higher even though the workload is finished in lesser number of cycles. This is because we have multithreading overhead, eg. a set of register files, delay-buffers, etc., for every new thread that we support and their increased power consumption offsets the gain in finishing the workload earlier. However, Figure 5.10 shows that translating the gain in IPC to scale down the frequency reduces the energy consumption as the number of threads increases. Slowing the processor more than compensates the increased power consumption in the multithreading hardware. At an operating frequency of 95MHz for 4-way multithreading, there is a reduction in the energy consumption by a factor of 1.8, on an average. Scaling the frequency down allows us to operate the processor at lower voltage levels. Figure 5.11 shows that this leads to further savings in energy as we increase the number of threads and reduce the operating voltage and frequency. Energy consumption reduces by a factor of 5.74, on an average, for a 4 thread SMT configuration. Figure 5.9, Figure 5.10 and

Figure 5.11 show the energy consumption data normalised to the highest value in each of the data sets.

Chapter 6

Conclusions

6.1 Summary

With a wide range of experiments, we have shown that our *extended split-issue* mechanism provides the VLIW processor with two very powerful capabilities. It makes the hardware capable of supporting simultaneous multithreading and presents the hardware designer the freedom to decouple the hardware implementation from the ISA. The *extended split-issue* mechanism therefore expands the hardware design-space in two dimensions. On one hand, we could decide to make the hardware capable of supporting SMT and on the other, we have the ability to choose to have an optimal number hardware functional units, thereby increasing hardware efficiency in novel ways. The extent of freedom provided by this mechanism to the hardware designer is unprecedented in the VLIW arena.

Using a commercial VLIW DSP architecture, TMS320C6201, we show that

incorporating SMT can reap rich benefits, either in terms of processing throughput or reduction in energy consumption. The average gain in IPC, over the base processor, was seen to be 120% for our SMT VLIW processor supporting 4 threads. It was also shown that such gains in throughput could be traded-off for major reduction in enery consumption. An average reduction in energy consumption by a factor of 5.74 was shown for a 4 threaded SMT, when the gains in IPC were translated to lower operating frequency and voltage. This is extremely paramount in the DSP area where real-time requirements make minimization of energy consumption more important than gains in throughput.

We have also shown that the efficiency of the VLIW DSP can be increased by choosing to have optimal number of functional units of each type based on their utilisation statistics. Results for different hardware configurations present us with further increased processor throughputs over the multithreaded processors, thereby proffering us with a wider variety of choices to build an efficient VLIW DSP without modifying the compiler or ISA.

6.2 Future Work

We briefly outline a few directions, in the expanded design-space presented by our work, that can be further explored to investigate the behaviour of our SMT

VLIW.

1. A further detailed exploration of the design-space, varying the set of hardware functional units to gain a thorough understanding of the trade-offs involved.

2. Formulating and investigating different issue policies other than the fixed priority-based scheme studied in this work. It would be interesting to extend the study of soft real-time scheduling [24] in the context of our design.

3. It would also be interesting to study the usage of different cache configurations in a multithreaded VLIW real-time scenario.

4. Estimating the cost of introducing the *extended split-issue* mechanism in terms of die-area.

Appendix A

Code Profiles

Here, we list the code profiles generated for 3 sample benchmarks and assembly code for the most commonly executed functions in these benchmarks.

A.1 Encrypt

Table A.1 shows the function profile for the Encrypt benchmark. It lists the data for 10 functions where the application spends most of its execution time.

The columnns of the table can be described as follows:

- % time : the percentage of the total running time of the program used by this function.
- cumulative seconds : a running sum of the number of seconds accounted for by this function and those listed above it.
- self seconds : the number of seconds accounted for by this function alone. This is the major sort for this listing.

% time	cumulative seconds	self seconds	calls	self us/call	total us/call	name
50.00	0.03	0.03	616	48.70	48.70	gfInvert
33.33	0.05	0.02	1231	16.25	16.25	gfMultiply
16.67	0.06	0.01	1331	7.51	7.51	gfSquare
0.00	0.06	0.00	5743	0.00	0.00	squareEncrypt
0.00	0.06	0.00	3697	0.00	0.00	gfClear
0.00	0.06	0.00	3125	0.00	0.00	gfAdd
0.00	0.06	0.00	1468	0.00	0.00	SHA1Transform
0.00	0.06	0.00	928	0.00	0.00	vlTakeBit
0.00	0.06	0.00	708	0.00	0.00	gfCopy
0.00	0.06	0.00	462	0.00	96.22	ecDouble

Table A.1: Function Profile for Encrypt

- calls : the number of times this function was invoked, if this function is profiled, else blank.

- self us/call : the average number of microseconds spent in this function per call, if this function is profiled, else blank.

- total us/call : the average number of milliseconds spent in this function and its descendents per call, if this function is profiled, else blank.

- name : the name of the function. This is the minor sort for this listing. The index shows the location of the function in the gprof listing. If the index is in parenthesis it shows where it would appear in the gprof listing if it were to be printed.

The assembly language listings for 3 most frequently executed functions are

given below.

```
.sect ".text"
.global _gfInvert

;******************************************************************************
;* FUNCTION NAME: _gfInvert                                                   *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,B0,B1,B2,B3,B4,*
;*                           B5,B6,B7,B8,B9,B10,B11,B12,B13,SP                *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,B0,B1,B2,B3,B4,*
;*                           B5,B6,B7,B8,B9,B10,B11,B12,B13,DP,SP             *
;*   Local Frame Size  : 0 Args + 216 Auto + 28 Save = 244 byte               *
;******************************************************************************
_gfInvert:
;** --------------------------------------------------------------------------*
           LDW     .D2T2   *+DP(_logt),B0    ; |388|
           ADDK    .S2     -248,SP           ; |381|
           NOP             2
           STW     .D2T2   B3,*+SP(232)      ; |381|

   [ B0]   B       .S1     L80               ; |388|
||         STW     .D2T1   A11,*+SP(228)     ; |381|

           STW     .D2T1   A10,*+SP(224)     ; |381|
           STW     .D2T2   B12,*+SP(244)     ; |381|
           STW     .D2T2   B11,*+SP(240)     ; |381|

           STW     .D2T2   B10,*+SP(236)     ; |381|
||         MVKL    .S2     RL80,B3           ; |388|

           MVKH    .S2     RL80,B3           ; |388|
||         MV      .L1X    B4,A11            ;
||         MV      .S1     A4,A10            ;
||         STW     .D2T2   B13,*+SP(248)     ; |381|

           ; BRANCH OCCURS                   ; |388|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |388|
           MVKL    .S1     SL22+0,A4         ; |388|
           MVKH    .S1     SL22+0,A4         ; |388|
           NOP             3
RL80:      ; CALL OCCURS                     ; |388|
;** --------------------------------------------------------------------------*
L80:
           LDW     .D2T2   *+DP(_expt),B0    ; |388|
           NOP             4
   [ B0]   B       .S1     L81               ; |388|
   [ B0]   MV      .L1     A10,A1
```

```
           NOP             4
           ; BRANCH OCCURS                 ; |388|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg     ; |388|
           MVKL    .S1     SL22+0,A4       ; |388|
           MVKL    .S2     RL82,B3         ; |388|
           MVKH    .S1     SL22+0,A4       ; |388|
           MVKH    .S2     RL82,B3         ; |388|
           NOP             1
RL82:      ; CALL OCCURS                   ; |388|
           MV      .L1     A10,A1
;** --------------------------------------------------------------------------*
L81:
   [ A1]   B       .S1     L82             ; |389|
   [ A1]   MV      .L1     A11,A1
           NOP             4
           ; BRANCH OCCURS                 ; |389|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg     ; |389|
           MVKL    .S1     SL23+0,A4       ; |389|
           MVKL    .S2     RL84,B3         ; |389|
           MVKH    .S1     SL23+0,A4       ; |389|
           MVKH    .S2     RL84,B3         ; |389|
           NOP             1
RL84:      ; CALL OCCURS                   ; |389|
           MV      .L1     A11,A1
;** --------------------------------------------------------------------------*
L82:
   [ A1]   B       .S1     L83             ; |390|
   [ A1]   MV      .L2X    A11,B4
   [ A1]   CMPEQ   .L2X    A10,B4,B0       ; |391|
           NOP             3
           ; BRANCH OCCURS                 ; |390|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg     ; |390|
           MVKL    .S1     SL24+0,A4       ; |390|
           MVKL    .S2     RL86,B3         ; |390|
           MVKH    .S1     SL24+0,A4       ; |390|
           MVKH    .S2     RL86,B3         ; |390|
           NOP             1
RL86:      ; CALL OCCURS                   ; |390|
           MV      .L2X    A11,B4
           CMPEQ   .L2X    A10,B4,B0       ; |391|
;** --------------------------------------------------------------------------*
L83:
   [!B0]   B       .S1     L84             ; |391|
   [!B0]   MV      .L2X    A11,B4
   [!B0]   LDHU    .D2T2   *B4,B0          ; |392|
           NOP             3
           ; BRANCH OCCURS                 ; |391|
;** --------------------------------------------------------------------------*
```

```
           B       .S1     __abort_msg       ; |391|
           MVKL    .S1     SL25+0,A4         ; |391|
           MVKL    .S2     RL88,B3           ; |391|
           MVKH    .S1     SL25+0,A4         ; |391|
           MVKH    .S2     RL88,B3           ; |391|
           NOP             1
RL88:      ; CALL OCCURS                     ; |391|
           MV      .L2X    A11,B4
           LDHU    .D2T2   *B4,B0            ; |392|
           NOP             3
;** --------------------------------------------------------------------------*
L84:
           NOP             1
   [!B0]   B       .S1     L92               ; |394|
           MVKL    .S2     0x10012,B11       ; |402|
           MVK     .S1     0x1,A0            ; |394|
           MVK     .S2     0x1,B10           ; |398|
           ADDAW   .D2     SP,19,B5          ; |400|

           ZERO    .L2     B5                ; |399|
||         MVKH    .S2     0x10012,B11       ; |402|
||         MV      .L1X    B5,A4             ; |400|
||         MVK     .S1     148,A11           ; |401|

           ; BRANCH OCCURS                   ; |394|
;** --------------------------------------------------------------------------*
           B       .S1     _gfCopy           ; |400|
           STH     .D1T2   B10,*A10          ; |398|
           STH     .D1T2   B10,*+A10(2)      ; |398|
           MVKL    .S2     RL90,B3           ; |400|
           MVKH    .S2     RL90,B3           ; |400|
           STH     .D2T2   B5,*+SP(4)        ; |399|
RL90:      ; CALL OCCURS                     ; |400|
           B       .S1     _gfClear          ; |401|
           MVKL    .S2     RL92,B3           ; |401|
           ADD     .L1X    A11,SP,A4         ;
           MVKH    .S2     RL92,B3           ; |401|
           NOP             2
RL92:      ; CALL OCCURS                     ; |401|
           LDHU    .D2T2   *+SP(76),B4       ; |402|
           MVK     .S2     0x7fff,B12
           NOP             3
           CMPEQ   .L2     B4,1,B0           ; |402|
   [ B0]   B       .S1     L90               ; |402|
           STW     .D2T2   B11,*+SP(148)     ; |402|
           STH     .D2T2   B10,*+SP(156)     ; |402|
           STH     .D2T2   B10,*+SP(184)     ; |402|
   [ B0]   LDHU    .D2T2   *+SP(78),B0       ; |406|
           NOP             1
           ; BRANCH OCCURS                   ; |402|
;** --------------------------------------------------------------------------*
```

```
           LDHU    .D2T2   *+SP(148),B4      ; |413|
           LDHU    .D2T2   *+SP(76),B5       ; |413|
           NOP             2
;** --------------------------------------------------------------------------*
L85:
           NOP             2
           CMPLT   .L2     B5,B4,B0          ; |413|
   [!B0]   B       .S1     L89               ; |413|

   [!B0]   LDHU    .D2T2   *+SP(76),B7       ; |417|
|| [ B0]   MVK     .S2     0xffff8001,B13

   [!B0]   LDHU    .D2T2   *+SP(148),B8      ; |417|
   [!B0]   ADDAW   .D2     SP,19,B6          ; |417|
   [!B0]   LDW     .D2T2   *+DP(_logt),B4    ; |417|
   [!B0]   MVK     .S2     148,B5            ; |417|
           ; BRANCH OCCURS                   ; |413|
;** --------------------------------------------------------------------------*
           LDHU    .D2T2   *+SP(76),B5       ; |438|
           LDHU    .D2T2   *+SP(148),B7      ; |438|
           NOP             1
;** --------------------------------------------------------------------------*
L86:
           ADDAW   .D2     SP,19,B6          ; |438|

           LDW     .D2T2   *+DP(_logt),B4    ; |438|
||         MVK     .S2     148,B8            ; |438|

           ADD     .L2     B8,SP,B5          ; |438|
||         LDHU    .D2T2   *+B6[B5],B6       ; |438|

           LDHU    .D2T2   *+B5[B7],B5       ; |438|
           NOP             3
           LDHU    .D2T2   *+B4[B6],B6       ; |438|
           LDHU    .D2T2   *+B4[B5],B4       ; |438|
           LDW     .D2T2   *+DP(_expt),B7    ; |439|
           LDHU    .D2T2   *+SP(148),B5      ; |437|
           MVK     .S1     32767,A0          ; |439|
           NOP             1
           SUB     .L2     B4,B6,B4          ; |438|
           ADD     .L2     B12,B4,B4         ; |438|
           EXTU    .S2     B4,16,16,B4       ; |438|

           LDHU    .D2T2   *+SP(76),B6       ; |437|
||         CMPLTU  .L1X    B4,A0,A1          ; |439|

   [!A1]   ADD     .L2     B13,B4,B4         ; |439|

           B       .S1     _gfAddMul         ; |440|
||         LDHU    .D2T2   *+B7[B4],B10      ; |439|
```

```
           MVK     .S1     148,A0            ; |440|
           MVKL    .S2     RL94,B3           ; |440|
           SUB     .L2     B5,B6,B4          ; |437|

           EXTU    .S2     B4,16,16,B11      ; |437|
||         ADD     .L1X    A0,SP,A4          ;

           MV      .L2     B10,B4            ; |440|
||         MV      .L1X    B11,A6            ; |440|
||         ADDAW   .D2     SP,19,B6          ; |440|
||         MVKH    .S2     RL94,B3           ; |440|

RL94:      ; CALL OCCURS                     ; |440|
           B       .S1     _gfAddMul         ; |441|
           MV      .L1X    B11,A6            ; |441|
           ADD     .L2     4,SP,B5           ; |441|
           MVKL    .S2     RL96,B3           ; |441|
           MV      .L1X    B5,A4             ; |441|

           MV      .D2     B10,B4            ; |441|
||         MV      .L2X    A10,B6            ; |441|
||         MVKH    .S2     RL96,B3           ; |441|

RL96:      ; CALL OCCURS                     ; |441|
           LDHU    .D2T2   *+SP(148),B4      ; |442|
           NOP             4
           CMPEQ   .L2     B4,1,B0           ; |442|
   [!B0]   B       .S1     L88               ; |442|
   [!B0]   LDHU    .D2T2   *+SP(76),B4       ; |434|
   [!B0]   LDHU    .D2T2   *+SP(148),B5      ; |434|
   [ B0]   LDHU    .D2T2   *+SP(150),B0      ; |426|
           NOP             2
           ; BRANCH OCCURS                   ; |442|
;** --------------------------------------------------------------------------*
           NOP             2
   [ B0]   B       .S1     L87               ; |426|
           MVKL    .S1     SL26+0,A4         ; |426|
           MVKL    .S2     RL98,B3           ; |426|
           MVKH    .S1     SL26+0,A4         ; |426|
           MVKH    .S2     RL98,B3           ; |426|
           NOP             1
           ; BRANCH OCCURS                   ; |426|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |426|
           NOP             5
RL98:      ; CALL OCCURS                     ; |426|
;** --------------------------------------------------------------------------*
L87:
           B       .S1     _gfSmallDiv       ; |427|
           LDHU    .D2T2   *+SP(150),B4      ;
           MVKL    .S2     RL100,B3          ; |427|
```

```
           ADD     .L2     4,SP,B5           ; |427|
           MVKH    .S2     RL100,B3          ; |427|
           MV      .L1X    B5,A4             ; |427|
RL100:     ; CALL OCCURS                     ; |427|
           B       .S1     _gfCopy           ; |428|
           MVKL    .S2     RL102,B3          ; |428|
           MV      .L1     A10,A4            ; |428|
           ADD     .L2     4,SP,B4           ; |428|
           MVKH    .S2     RL102,B3          ; |428|
           NOP             1
RL102:     ; CALL OCCURS                     ; |428|
           B       .S1     _gfClear          ; |430|
           ADD     .L2     4,SP,B4           ; |430|
           MVKL    .S2     RL104,B3          ; |430|
           MV      .L1X    B4,A4             ; |430|
           MVKH    .S2     RL104,B3          ; |430|
           NOP             1
RL104:     ; CALL OCCURS                     ; |430|
           B       .S1     _gfClear          ; |430|
           ADDAW   .D2     SP,19,B4          ; |430|
           MVKL    .S2     RL106,B3          ; |430|
           MV      .L1X    B4,A4             ; |430|
           MVKH    .S2     RL106,B3          ; |430|
           NOP             1
RL106:     ; CALL OCCURS                     ; |430|
           B       .S1     _gfClear          ; |430|
           MVK     .S1     148,A0            ; |430|
           MVKL    .S2     RL108,B3          ; |430|
           MVKH    .S2     RL108,B3          ; |430|
           ADD     .L1X    A0,SP,A4          ;
           NOP             1
RL108:     ; CALL OCCURS                     ; |430|
           B       .S1     L92               ; |431|
           ZERO    .L1     A0                ; |410|
           NOP             4
           ; BRANCH OCCURS                   ; |431|
;** --------------------------------------------------------------------------*
L88:
           NOP             2
           CMPLT   .L2     B5,B4,B0          ; |434|
   [!B0]   B       .S1     L86               ; |434|
   [!B0]   LDHU    .D2T2   *+SP(76),B5       ; |438|
   [!B0]   LDHU    .D2T2   *+SP(148),B7      ; |438|
   [ B0]   LDHU    .D2T2   *+SP(76),B7       ; |417|
   [ B0]   LDHU    .D2T2   *+SP(148),B8      ; |417|
           NOP             1
           ; BRANCH OCCURS                   ; |434|
;** --------------------------------------------------------------------------*
           ADDAW   .D2     SP,19,B6          ; |417|

           LDW     .D2T2   *+DP(_logt),B4    ; |417|
```

```
||         MVK     .S2     148,B5            ; |417|

;** --------------------------------------------------------------------------*
L89:

           LDHU    .D2T2   *+B6[B7],B5       ; |417|
||         ADD     .L2     B5,SP,B6          ; |417|

           LDHU    .D2T2   *+B6[B8],B6       ; |417|
           NOP             3
           LDHU    .D2T2   *+B4[B5],B5       ; |417|
           LDHU    .D2T2   *+B4[B6],B4       ; |417|
           LDHU    .D2T2   *+SP(76),B7       ; |416|
           LDHU    .D2T2   *+SP(148),B6      ; |416|
           MVK     .S1     32767,A0          ; |418|
           MVKL    .S2     RL110,B3          ; |419|
           SUB     .L2     B5,B4,B4          ; |417|

           LDW     .D2T2   *+DP(_expt),B5    ; |418|
||         ADD     .L2     B12,B4,B4         ; |417|

           EXTU    .S2     B4,16,16,B4       ; |417|
           CMPLTU  .L1X    B4,A0,A1          ; |418|
   [!A1]   MVK     .S1     32767,A0          ; |418|
   [!A1]   SUB     .L2X    B4,A0,B4          ; |418|

           B       .S1     _gfAddMul         ; |419|
||         LDHU    .D2T2   *+B5[B4],B11      ; |418|

           SUB     .L2     B7,B6,B5          ; |416|
           EXTU    .S2     B5,16,16,B10      ; |416|
           MVK     .S2     148,B4            ; |419|

           ADD     .L2     B4,SP,B6          ; |419|
||         MV      .L1X    B10,A6            ; |419|
||         ADDAW   .D2     SP,19,B4          ; |419|

           MV      .L2     B11,B4            ; |419|
||         MV      .L1X    B4,A4             ; |419|
||         MVKH    .S2     RL110,B3          ; |419|

RL110:     ; CALL OCCURS                     ; |419|
           B       .S1     _gfAddMul         ; |420|
           MVKL    .S2     RL112,B3          ; |420|
           MVKH    .S2     RL112,B3          ; |420|
           ADD     .L2     4,SP,B6           ; |420|
           MV      .D2     B11,B4            ; |420|

           MV      .L1X    B10,A6            ; |420|
||         MV      .S1     A10,A4            ; |420|
```

```
RL112:     ; CALL OCCURS                     ; |420|
           LDHU    .D2T2   *+SP(76),B4       ; |421|
           NOP             4
           CMPEQ   .L2     B4,1,B0           ; |421|
   [!B0]   B       .S1     L85               ; |421|
   [!B0]   LDHU    .D2T2   *+SP(148),B4      ; |413|
   [!B0]   LDHU    .D2T2   *+SP(76),B5       ; |413|
   [ B0]   LDHU    .D2T2   *+SP(78),B0       ; |406|
           NOP             2
           ; BRANCH OCCURS                   ; |421|
;** --------------------------------------------------------------------------*
L90:
           NOP             3
   [ B0]   B       .S1     L91               ; |406|
           MVKL    .S1     SL27+0,A4         ; |406|
           MVKL    .S2     RL114,B3          ; |406|
           MVKH    .S1     SL27+0,A4         ; |406|
           MVKH    .S2     RL114,B3          ; |406|
           NOP             1
           ; BRANCH OCCURS                   ; |406|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |406|
           NOP             5
RL114:     ; CALL OCCURS                     ; |406|
;** --------------------------------------------------------------------------*
L91:
           B       .S1     _gfSmallDiv       ; |407|
           LDHU    .D2T2   *+SP(78),B4       ;
           MVKL    .S2     RL116,B3          ; |407|
           MVKH    .S2     RL116,B3          ; |407|
           MV      .L1     A10,A4            ; |407|
           NOP             1
RL116:     ; CALL OCCURS                     ; |407|
           B       .S1     _gfClear          ; |409|
           ADD     .L2     4,SP,B4           ; |409|
           MVKL    .S2     RL118,B3          ; |409|
           MV      .L1X    B4,A4             ; |409|
           MVKH    .S2     RL118,B3          ; |409|
           NOP             1
RL118:     ; CALL OCCURS                     ; |409|
           B       .S1     _gfClear          ; |409|
           ADDAW   .D2     SP,19,B4          ; |409|
           MVKL    .S2     RL120,B3          ; |409|
           MV      .L1X    B4,A4             ; |409|
           MVKH    .S2     RL120,B3          ; |409|
           NOP             1
RL120:     ; CALL OCCURS                     ; |409|
           B       .S1     _gfClear          ; |409|
           MVKL    .S2     RL122,B3          ; |409|
           MVK     .S1     148,A0            ; |409|
           ADD     .L1X    A0,SP,A4          ;
```

```
           MVKH    .S2     RL122,B3          ; |409|
           NOP             1
RL122:     ; CALL OCCURS                     ; |409|
           ZERO    .L1     A0                ; |410|
;** --------------------------------------------------------------------------*
L92:
           LDW     .D2T2   *+SP(232),B3      ; |443|
           LDW     .D2T2   *+SP(244),B12     ; |443|
           LDW     .D2T2   *+SP(240),B11     ; |443|
           LDW     .D2T2   *+SP(236),B10     ; |443|
           LDW     .D2T1   *+SP(228),A11     ; |443|

           B       .S2     B3                ; |443|
||         LDW     .D2T2   *+SP(248),B13     ; |443|

           LDW     .D2T1   *+SP(224),A10     ; |443|
           ADDK    .S2     248,SP            ; |443|
           MV      .L1     A0,A4             ; |410|
           NOP             2
           ; BRANCH OCCURS                   ; |443|

.sect ".text"
.global _gfMultiply

;******************************************************************************
;* FUNCTION NAME: _gfMultiply                                                 *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,SP             *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,DP,SP          *
;*   Local Frame Size  : 0 Args + 40 Auto + 28 Save = 68 byte                 *
;******************************************************************************
_gfMultiply:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B3,*SP--(72)      ; |268|
           STW     .D2T1   A14,*+SP(64)      ; |268|
           LDW     .D2T1   *+DP(_logt),A14   ; |275|
           MVKL    .S2     RL40,B3           ; |275|
           MVKH    .S2     RL40,B3           ; |275|
           NOP             2
           MV      .L1     A14,A1            ; |275|
   [ A1]   B       .S1     L40               ; |275|
           STW     .D2T1   A15,*+SP(68)      ; |268|
           STW     .D2T1   A13,*+SP(60)      ; |268|
           STW     .D2T1   A10,*+SP(48)      ; |268|
           STW     .D2T1   A11,*+SP(52)      ; |268|

           MV      .L1     A6,A15            ;
||         MV      .S1X    B4,A13            ;
||         MV      .D1     A4,A10            ;
```

```
||         STW     .D2T1   A12,*+SP(56)      ; |268|

           ; BRANCH OCCURS                   ; |275|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |275|
           MVKL    .S1     SL10+0,A4         ; |275|
           MVKH    .S1     SL10+0,A4         ; |275|
           NOP             3
RL40:      ; CALL OCCURS                     ; |275|
;** --------------------------------------------------------------------------*
L40:
           LDW     .D2T2   *+DP(_expt),B0    ; |275|
           NOP             4
   [ B0]   B       .S1     L41               ; |275|
   [ B0]   MV      .L1     A13,A1
           NOP             4
           ; BRANCH OCCURS                   ; |275|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |275|
           MVKL    .S1     SL10+0,A4         ; |275|
           MVKL    .S2     RL42,B3           ; |275|
           MVKH    .S1     SL10+0,A4         ; |275|
           MVKH    .S2     RL42,B3           ; |275|
           NOP             1
RL42:      ; CALL OCCURS                     ; |275|
           MV      .L1     A13,A1
;** --------------------------------------------------------------------------*
L41:
   [ A1]   B       .S1     L42               ; |276|
   [ A1]   MV      .L1     A15,A1
           NOP             4
           ; BRANCH OCCURS                   ; |276|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |276|
           MVKL    .S1     SL11+0,A4         ; |276|
           MVKL    .S2     RL44,B3           ; |276|
           MVKH    .S1     SL11+0,A4         ; |276|
           MVKH    .S2     RL44,B3           ; |276|
           NOP             1
RL44:      ; CALL OCCURS                     ; |276|
           MV      .L1     A15,A1
;** --------------------------------------------------------------------------*
L42:
   [ A1]   B       .S1     L43               ; |277|
   [ A1]   CMPEQ   .L1     A10,A13,A1        ; |278|
           NOP             4
           ; BRANCH OCCURS                   ; |277|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |277|
           MVKL    .S1     SL12+0,A4         ; |277|
           MVKL    .S2     RL46,B3           ; |277|
```

```
           MVKH    .S1     SL12+0,A4         ; |277|
           MVKH    .S2     RL46,B3           ; |277|
           NOP             1
RL46:      ; CALL OCCURS                     ; |277|
           CMPEQ   .L1     A10,A13,A1        ; |278|
;** --------------------------------------------------------------------------*
L43:
   [!A1]   B       .S1     L44               ; |278|
   [!A1]   CMPEQ   .L1     A10,A15,A1        ; |279|
           NOP             4
           ; BRANCH OCCURS                   ; |278|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |278|
           MVKL    .S1     SL13+0,A4         ; |278|
           MVKL    .S2     RL48,B3           ; |278|
           MVKH    .S1     SL13+0,A4         ; |278|
           MVKH    .S2     RL48,B3           ; |278|
           NOP             1
RL48:      ; CALL OCCURS                     ; |278|
           CMPEQ   .L1     A10,A15,A1        ; |279|
;** --------------------------------------------------------------------------*
L44:
   [!A1]   B       .S1     L45               ; |279|
   [!A1]   LDHU    .D1T1   *A13,A12          ; |280|
           NOP             4
           ; BRANCH OCCURS                   ; |279|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |279|
           MVKL    .S1     SL14+0,A4         ; |279|
           MVKL    .S2     RL50,B3           ; |279|
           MVKH    .S1     SL14+0,A4         ; |279|
           MVKH    .S2     RL50,B3           ; |279|
           NOP             1
RL50:      ; CALL OCCURS                     ; |279|
           LDHU    .D1T1   *A13,A12          ; |280|
           NOP             4
;** --------------------------------------------------------------------------*
L45:
           MV      .L1     A12,A1
   [!A1]   B       .S1     L46               ; |280|
   [ A1]   LDHU    .D1T1   *A15,A11          ; |280|
           NOP             4
           ; BRANCH OCCURS                   ; |280|
;** --------------------------------------------------------------------------*
           MV      .L1     A11,A1
   [ A1]   B       .S1     L47               ; |280|
           MV      .L1     A11,A0
           ADDAH   .D1     A15,A0,A5
           ADD     .L1X    4,SP,A3

           ADDAH   .D1     A3,A0,A3
```

```
||         MV      .L2X    A11,B0

   [ A1]   MV      .L1X    B0,A1
|| [ A1]   MV      .D1     A14,A4
|| [ A1]   MVK     .S1     0x1,A2          ; init prolog collapse predicate

           ; BRANCH OCCURS                 ; |280|
;** --------------------------------------------------------------------------*
L46:
           B       .S1     L54             ; |302|
           ZERO    .L1     A0              ; |302|
           STH     .D1T1   A0,*A10         ; |302|
           NOP             3
           ; BRANCH OCCURS                 ; |302|
;** --------------------------------------------------------------------------*
L47:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L48:    ; PIPED LOOP KERNEL

   [ B0]   B       .S2     L48             ; |284|
|| [!A2]   LDHU    .D1T1   *+A4[A0],A6     ;  ^ |283|

   [ A1]   LDHU    .D1T1   *A5--,A0        ; @|283|
           NOP             3

   [ A2]   SUB     .S1     A2,1,A2         ;
|| [ A1]   SUB     .L1     A1,1,A1         ;
|| [!A2]   STH     .D1T1   A6,*A3--        ;  ^ |283|
|| [ B0]   SUB     .L2     B0,1,B0         ; @|284|

;** --------------------------------------------------------------------------*
L49:    ; PIPED LOOP EPILOG
;** --------------------------------------------------------------------------*
           B       .S1     _gfClear        ; |286|
           MVKL    .S2     RL52,B3         ; |286|
           MVKH    .S2     RL52,B3         ; |286|
           MV      .L1     A10,A4          ; |286|
           NOP             2
RL52:      ; CALL OCCURS                   ; |286|
           MV      .L1     A12,A0
           ADDAH   .D1     A13,A0,A3

           ADD     .L1     2,A3,A8
||         SHL     .S1     A0,1,A6
||         LDW     .D2T1   *+DP(_logt),A9
||         MV      .L2X    A12,B1          ; |287|

           LDHU    .D1T1   *--A8,A0        ; |288|
           NOP             4
;** --------------------------------------------------------------------------*
L50:
```

```
           LDHU    .D1T1   *+A9[A0],A7       ; |288|
           MVK     .S1     32767,A0         ; |288|
           NOP             3
           CMPEQ   .L1     A7,A0,A1         ; |288|
   [ A1]   B       .S1     L53              ; |288|
   [ A1]   SUB     .L2     B1,1,B1          ; |296|
   [!A1]   ADD     .L1X    4,SP,A3
   [!A1]   MV      .S1     A11,A0           ; |289|
   [!A1]   ADDAH   .D1     A3,A0,A3

   [!A1]   SHL     .S1     A0,1,A3          ; |289|
|| [!A1]   ADD     .L1     2,A3,A4
|| [!A1]   MV      .L2X    A11,B0           ; |290|

           ; BRANCH OCCURS                  ; |288|
;** --------------------------------------------------------------------------*
           LDHU    .D1T1   *--A4,A0         ; |290|
           MVK     .S1     32767,A5         ; |290|
           NOP             3
;** --------------------------------------------------------------------------*
L51:
           CMPEQ   .L1     A0,A5,A1         ; |290|
   [ A1]   B       .S1     L52              ; |290|
   [ A1]   SUB     .L2     B0,1,B0          ; |294|
   [!A1]   LDW     .D2T2   *+DP(_expt),B5
   [!A1]   ADD     .L1     A7,A0,A0         ; |292|
   [!A1]   EXTU    .S1     A0,16,16,A0      ; |292|

   [!A1]   ADD     .S1     A6,A3,A5
|| [!A1]   CMPLTU  .L1     A0,A5,A1         ; |292|

           ; BRANCH OCCURS                  ; |290|
;** --------------------------------------------------------------------------*

   [ A1]   MV      .L2X    A0,B4            ; |292|
||         ADD     .L1     A10,A5,A5
|| [!A1]   MVK     .S2     32767,B4         ; |292|

           SUB     .L1     A5,2,A0
|| [!A1]   SUB     .L2X    A0,B4,B4         ; |292|

           LDHU    .D1T1   *A0,A5           ; |292|
||         LDHU    .D2T2   *+B5[B4],B4      ; |292|

           SUB     .L2     B0,1,B0          ; |294|
           NOP             3
           XOR     .L1X    B4,A5,A5         ; |292|
           STH     .D1T1   A5,*A0           ; |292|
;** --------------------------------------------------------------------------*
L52:
   [ B0]   B       .S1     L51              ; |294|
```

```
           SUB     .L1     A3,2,A3           ; |294|
|| [ B0]   LDHU    .D1T1   *--A4,A0          ; |290|
|| [!B0]   SUB     .L2     B1,1,B1           ; |296|

   [ B0]   MVK     .S1     32767,A5          ; |290|
           NOP             3
           ; BRANCH OCCURS                   ; |294|
;** --------------------------------------------------------------------------*
L53:
   [ B1]   B       .S1     L50               ; |296|

           SUB     .L1     A6,2,A6           ; |296|
|| [ B1]   LDHU    .D1T1   *--A8,A0          ; |288|

           NOP             4
           ; BRANCH OCCURS                   ; |296|
;** --------------------------------------------------------------------------*
           B       .S1     _gfReduce         ; |299|
           ADD     .L1     A11,A12,A0        ;
           SUB     .L1     A0,1,A0           ; |297|
           MVKL    .S2     RL54,B3           ; |299|
           STH     .D1T1   A0,*A10           ; |297|

           MVKH    .S2     RL54,B3           ; |299|
||         MV      .L1     A10,A4            ; |299|

RL54:      ; CALL OCCURS                     ; |299|
;** --------------------------------------------------------------------------*
L54:
           B       .S1     _memset           ; |306|
           MVKL    .S2     RL56,B3           ; |306|
           ADD     .L2     4,SP,B5           ; |306|
           MVKH    .S2     RL56,B3           ; |306|
           MV      .L1X    B5,A4             ; |306|

           MVK     .S1     0x26,A6           ; |306|
||         ZERO    .L2     B4                ; |306|

RL56:      ; CALL OCCURS                     ; |306|
           LDW     .D2T1   *+SP(68),A15      ; |307|
           LDW     .D2T1   *+SP(64),A14      ; |307|
           LDW     .D2T1   *+SP(60),A13      ; |307|
           LDW     .D2T1   *+SP(56),A12      ; |307|
           LDW     .D2T1   *+SP(52),A11      ; |307|
           LDW     .D2T1   *+SP(48),A10      ; |307|
           LDW     .D2T2   *++SP(72),B3      ; |307|
           NOP             4
           B       .S2     B3                ; |307|
           NOP             5
           ; BRANCH OCCURS                   ; |307|
```

```
.sect ".text"
.global _gfSquare

;******************************************************************************
;* FUNCTION NAME: _gfSquare                                                   *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,B0,B1, *
;*                           B2,B3,B4,B5,B6,B7,B8,B9,SP                       *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,B0,B1, *
;*                           B2,B3,B4,B5,B6,B7,B8,B9,DP,SP                    *
;*   Local Frame Size  : 0 Args + 0 Auto + 20 Save = 20 byte                  *
;******************************************************************************
_gfSquare:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B3,*SP--(24)      ; |310|
           STW     .D2T1   A12,*+SP(16)      ; |310|
           LDW     .D2T1   *+DP(_logt),A12   ; |316|
           MVKL    .S2     RL58,B3           ; |316|
           MVKH    .S2     RL58,B3           ; |316|
           NOP             2
           MV      .L1     A12,A1            ; |316|
   [ A1]   B       .S1     L55               ; |316|
           STW     .D2T1   A13,*+SP(20)      ; |310|
           STW     .D2T1   A11,*+SP(12)      ; |310|

           MV      .L1X    B4,A13            ;
||         MV      .S1     A4,A11            ;
||         STW     .D2T1   A10,*+SP(8)       ; |310|

   [ A1]   LDW     .D2T1   *+DP(_expt),A10   ;
           NOP             1
           ; BRANCH OCCURS                   ; |316|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |316|
           MVKL    .S1     SL15+0,A4         ; |316|
           MVKH    .S1     SL15+0,A4         ; |316|
           NOP             3
RL58:      ; CALL OCCURS                     ; |316|
           LDW     .D2T1   *+DP(_expt),A10   ;
           NOP             1
;** --------------------------------------------------------------------------*
L55:
           NOP             3
           MV      .L1     A10,A1
   [ A1]   B       .S1     L56               ; |316|
   [ A1]   MV      .L1     A11,A1
           NOP             4
           ; BRANCH OCCURS                   ; |316|
;** --------------------------------------------------------------------------*
```

```
           B       .S1     __abort_msg       ; |316|
           MVKL    .S1     SL15+0,A4         ; |316|
           MVKL    .S2     RL60,B3           ; |316|
           MVKH    .S1     SL15+0,A4         ; |316|
           MVKH    .S2     RL60,B3           ; |316|
           NOP             1
RL60:      ; CALL OCCURS                     ; |316|
           MV      .L1     A11,A1
;** --------------------------------------------------------------------------*
L56:
   [ A1]   B       .S1     L57               ; |317|
   [ A1]   MV      .L1     A13,A1
           NOP             4
           ; BRANCH OCCURS                   ; |317|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |317|
           MVKL    .S1     SL16+0,A4         ; |317|
           MVKL    .S2     RL62,B3           ; |317|
           MVKH    .S1     SL16+0,A4         ; |317|
           MVKH    .S2     RL62,B3           ; |317|
           NOP             1
RL62:      ; CALL OCCURS                     ; |317|
           MV      .L1     A13,A1
;** --------------------------------------------------------------------------*
L57:
   [ A1]   B       .S1     L58               ; |318|
   [ A1]   MV      .L2X    A13,B4
   [ A1]   LDHU    .D2T1   *B4,A2            ; |319|
           NOP             3
           ; BRANCH OCCURS                   ; |318|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |318|
           MVKL    .S1     SL17+0,A4         ; |318|
           MVKL    .S2     RL64,B3           ; |318|
           MVKH    .S1     SL17+0,A4         ; |318|
           MVKH    .S2     RL64,B3           ; |318|
           NOP             1
RL64:      ; CALL OCCURS                     ; |318|
           MV      .L2X    A13,B4
           LDHU    .D2T1   *B4,A2            ; |319|
           NOP             3
;** --------------------------------------------------------------------------*
L58:
           MVK     .S1     0x7fff,A5         ; |322|
   [!A2]   B       .S1     L63               ; |339|

   [!A2]   MV      .L1     A11,A3            ; |322|
||         ZERO    .D1     A0                ; |339|

   [!A2]   STH     .D1T1   A0,*A3            ; |339|
||         MV      .L1     A2,A4             ; |322|
```

```
||         MVK     .S1     32767,A6          ; |322|

   [ A2]   LDHU    .D1T1   *+A13[A2],A3      ; |322|
           NOP             2
           ; BRANCH OCCURS                   ; |339|
;** --------------------------------------------------------------------------*
           LDHU    .D1T1   *+A13[A2],A0      ; |322|
           NOP             1
           LDHU    .D1T1   *+A12[A3],A3      ; |322|
           NOP             4
           CMPEQ   .L1     A3,A6,A1          ; |322|

   [ A1]   B       .S1     L59               ; |325|
||         LDHU    .D1T1   *+A12[A0],A0      ; |322|

           ZERO    .L1     A3                ; |325|
           ADDAW   .D1     A11,A4,A4         ; |325|
   [ A1]   STH     .D1T1   A3,*-A4(2)        ; |325|
   [ A1]   SUB     .L2X    A2,1,B0           ; |327|
           SHL     .S1     A0,1,A0           ; |323|
           ; BRANCH OCCURS                   ; |325|
;** --------------------------------------------------------------------------*
           EXTU    .S1     A0,16,16,A0       ; |323|

           CMPLTU  .L1     A0,A5,A1          ; |323|
||         MV      .L2X    A0,B4             ; |323|
||         MVK     .S2     32767,B5          ; |323|

   [!A1]   SUB     .L2X    A0,B5,B4          ; |323|
           MV      .L2X    A10,B5            ; |323|
           LDHU    .D2T2   *+B5[B4],B4       ; |323|
           ADDAW   .D1     A11,A2,A0         ; |323|
           SUB     .L2X    A2,1,B0           ; |327|
           NOP             2
           STH     .D1T2   B4,*-A0(2)        ; |323|
;** --------------------------------------------------------------------------*
L59:
   [!B0]   B       .S1     L62               ; |327|
           MV      .L2X    A13,B4            ; |327|
           ADDAH   .D2     B4,B0,B4
           ADDAW   .D1     A11,A2,A0
           ADD     .L1X    2,B4,A3

           SUB     .S1     A0,6,A0
||         ZERO    .D1     A4

           ; BRANCH OCCURS                   ; |327|
;** --------------------------------------------------------------------------*
           LDHU    .D1T1   *--A3,A5          ; |329|
           STH     .D1T1   A4,*+A0(2)        ; |328|
           NOP             3
```

```
;** -----------------------------------------------------------------------*
L60:
           LDHU    .D1T1   *+A12[A5],A5      ; |329|
           MVK     .S1     32767,A6          ; |329|
           NOP             3
           CMPEQ   .L1     A5,A6,A1          ; |329|
   [ A1]   B       .S1     L61               ; |332|
   [ A1]   STH     .D1T1   A4,*A0            ; |332|
   [ A1]   SUB     .L2     B0,1,B0           ; |334|
   [!A1]   SHL     .S1     A5,1,A5           ; |330|

   [!A1]   EXTU    .S1     A5,16,16,A5       ; |330|
|| [!A1]   MVK     .S2     32767,B4          ; |330|

   [!A1]   MV      .L2X    A5,B4             ; |330|
|| [!A1]   CMPLTU  .L1X    A5,B4,A1          ; |330|

           ; BRANCH OCCURS                   ; |332|
;** -----------------------------------------------------------------------*

   [ A1]   MV      .L2X    A5,B4             ; |330|
|| [!A1]   MVK     .S2     32767,B5          ; |330|

           MV      .L2X    A10,B5            ; |330|
|| [!A1]   SUB     .S2     B4,B5,B4          ; |330|

           LDHU    .D2T2   *+B5[B4],B4       ; |330|
           SUB     .L2     B0,1,B0           ; |334|
           NOP             3
           STH     .D1T2   B4,*A0            ; |330|
;** -----------------------------------------------------------------------*
L61:
   [ B0]   B       .S1     L60               ; |334|

           SUB     .L1     A0,4,A0           ; |334|
|| [ B0]   LDHU    .D1T1   *--A3,A5          ; |329|

   [ B0]   STH     .D1T1   A4,*+A0(2)        ; |328|
           NOP             3
           ; BRANCH OCCURS                   ; |334|
;** -----------------------------------------------------------------------*
L62:
           B       .S1     _gfReduce         ; |337|
           SHL     .S1     A2,1,A0           ; |335|
           SUB     .L1     A0,1,A0           ; |335|
           MVKL    .S2     RL66,B3           ; |337|
           STH     .D1T1   A0,*A11           ; |335|

           MVKH    .S2     RL66,B3           ; |337|
||         MV      .L1     A11,A4            ; |337|
```

```
RL66:      ; CALL OCCURS                     ; |337|
;** --------------------------------------------------------------------------*
L63:
           LDW     .D2T1   *+SP(20),A13      ; |341|
           LDW     .D2T1   *+SP(16),A12      ; |341|
           LDW     .D2T1   *+SP(12),A11      ; |341|
           LDW     .D2T1   *+SP(8),A10       ; |341|
           LDW     .D2T2   *++SP(24),B3      ; |341|
           NOP             4
           B       .S2     B3                ; |341|
           NOP             5
           ; BRANCH OCCURS                   ; |341|
```

A.2 GSM_Encode

Table A.2 shows the function profile for the GSM_Encode benchmark. It lists the data for 10 functions where the application spends most of its execution time.

% time	cumulative seconds	self seconds	calls	self us/call	total us/call	name
50.00	0.01	0.01	133	75.19	75.19	Autocorrelation
50.00	0.02	0.01	133	75.19	150.38	gsm_encode
0.00	0.02	0.00	6916	0.00	0.00	gsm_asr
0.00	0.02	0.00	1315	0.00	0.00	gsm_norm
0.00	0.02	0.00	1231	0.00	0.00	gsm_mult
0.00	0.02	0.00	1064	0.00	0.00	gsm_div
0.00	0.02	0.00	1064	0.00	0.00	gsm_sub
0.00	0.02	0.00	532	0.00	0.00	APCM_quantization
0.00	0.02	0.00	532	0.00	0.00	Calculation_of_the_LTP_parameters
0.00	0.02	0.00	532	0.00	0.00	Gsm_Long_Term_Predictor

Table A.2: Function Profile for GSM_Encode

The columnns of the table can be described as follows:

- % time : the percentage of the total running time of the program used by this function.
- cumulative seconds : a running sum of the number of seconds accounted for by this function and those listed above it.
- self seconds : the number of seconds accounted for by this function alone. This is the major sort for this listing.
- calls : the number of times this function was invoked, if this function is profiled, else blank.
- self us/call : the average number of microseconds spent in this function per call, if this function is profiled, else blank.
- total us/call : the average number of milliseconds spent in this function and its descendents per call, if this function is profiled, else blank.
- name : the name of the function. This is the minor sort for this listing. The index shows the location of the function in the gprof listing. If the index is in parenthesis it shows where it would appear in the gprof listing if it were to be printed.

The assembly language listings for 2 most frequently executed functions are given below.

```
.sect ".text"

;******************************************************************************
;* FUNCTION NAME: _Autocorrelation                                            *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 4 Auto + 44 Save = 48 byte                  *
;******************************************************************************
_Autocorrelation:
;** --------------------------------------------------------------------------*
           STW     .D2T2   B13,*SP--(48)     ; |26|
           STW     .D2T1   A10,*+SP(8)       ; |26|
           STW     .D2T2   B12,*+SP(44)      ; |26|
           STW     .D2T2   B10,*+SP(36)      ; |26|
           STW     .D2T2   B3,*+SP(32)       ; |26|
           STW     .D2T1   A15,*+SP(28)      ; |26|
           STW     .D2T1   A14,*+SP(24)      ; |26|
           STW     .D2T1   A13,*+SP(20)      ; |26|
           STW     .D2T1   A12,*+SP(16)      ; |26|

           STW     .D2T1   A11,*+SP(12)      ; |26|
||         MVK     .S2     0xa0,B0           ; |48|

           ZERO    .L1     A10               ; |47|
||         MV      .L2     B4,B12            ;
||         STW     .D2T2   B11,*+SP(40)      ; |26|
||         MV      .S2X    A4,B13            ;
||         SUB     .S1     A4,2,A5           ;

           LDH     .D1T1   *++A5,A0          ; |49|
           NOP             3
;** --------------------------------------------------------------------------*
L1:
           NOP             1
           CMPLT   .L1     A0,0,A1           ; |49|
   [!A1]   B       .S1     L2                ; |49|
   [!A1]   MV      .L1     A0,A3             ; |49|
   [!A1]   SUB     .L2     B0,1,B0           ; |51|
   [ A1]   MVK     .S1     -32768,A3         ; |49|
   [ A1]   CMPEQ   .L1     A0,A3,A1          ; |49|
           NOP             1
           ; BRANCH OCCURS                   ; |49|
;** --------------------------------------------------------------------------*

   [ A1]   MVK     .S1     0x7fff,A3         ; |49|
|| [!A1]   NEG     .L1     A0,A0             ; |49|
```

```
   [!A1]   EXT     .S1     A0,16,16,A3       ; |49|
           SUB     .L2     B0,1,B0           ; |51|
;** --------------------------------------------------------------------------*
L2:
   [ B0]   B       .S1     L1                ; |51|
           CMPGT   .L1     A3,A10,A1         ; |50|

   [ A1]   MV      .L1     A3,A10            ;
|| [ B0]   LDH     .D1T1   *++A5,A0          ; |49|

   [!B0]   MV      .L1     A10,A2            ; |57|
           NOP             2
           ; BRANCH OCCURS                   ; |51|
;** --------------------------------------------------------------------------*
   [!A2]   B       .S1     L4                ; |55|
           ZERO    .L1     A0                ; |55|

           STW     .D2T1   A0,*+SP(4)        ; |57|
||         CMPGT   .L1     A10,0,A1          ; |57|

   [!A2]   LDW     .D2T1   *+SP(4),A0
           NOP             2
           ; BRANCH OCCURS                   ; |55|
;** --------------------------------------------------------------------------*
   [ A1]   B       .S1     L3                ; |57|
           MVKL    .S2     RL0,B3            ; |57|
           MVKH    .S2     RL0,B3            ; |57|
           NOP             3
           ; BRANCH OCCURS                   ; |57|
;** --------------------------------------------------------------------------*
           B       .S1     __abort_msg       ; |57|
           MVKL    .S1     SL1+0,A4          ; |57|
           MVKH    .S1     SL1+0,A4          ; |57|
           NOP             3
RL0:       ; CALL OCCURS                     ; |57|
;** --------------------------------------------------------------------------*
L3:
           B       .S1     _gsm_norm         ; |58|
           MVKL    .S2     RL2,B3            ; |58|
           MVKH    .S2     RL2,B3            ; |58|
           SHL     .S1     A10,16,A4         ; |58|
           NOP             2
RL2:       ; CALL OCCURS                     ; |58|
           SUB     .L1     4,A4,A0           ; |58|
           EXT     .S1     A0,16,16,A0       ; |58|
           STW     .D2T1   A0,*+SP(4)        ; |58|
           LDW     .D2T1   *+SP(4),A0
           NOP             2
;** --------------------------------------------------------------------------*
L4:
```

```
           NOP             2
           CMPGT   .L1     A0,0,A1           ; |79|
   [!A1]   B       .S1     L21               ; |79|

           CMPEQ   .L1     A0,1,A2           ;
||         MV      .L2X    A0,B5             ; |79|

   [!A1]   MV      .L1X    B12,A0            ; |100|
|| [!A1]   ZERO    .S1     A6                ; |109|

   [!A1]   STW     .D1T1   A6,*+A0(24)       ; |109|
|| [!A1]   MV      .L1X    B13,A5            ; |109|

   [!A1]   LDH     .D1T1   *A5,A0            ; |101|
   [!A1]   LDH     .D1T1   *A5,A3            ; |111|
           ; BRANCH OCCURS                   ; |79|
;** --------------------------------------------------------------------------*
   [ A2]   B       .S1     L16               ; |83|
           MVK     .S2     0x14,B0           ; |80|
           CMPEQ   .L2     B5,2,B1           ; |83|
   [ A2]   MVK     .S1     0x1,A2            ; init prolog collapse predicate
           MV      .L1X    B13,A0
           SUBAW   .D1     A0,4,A4
           ; BRANCH OCCURS                   ; |83|
;** --------------------------------------------------------------------------*
   [ B1]   B       .S1     L12               ; |83|
           SUBAW   .D1     A0,4,A5
           MVK     .S2     0x14,B4           ; |81|
           CMPEQ   .L2     B5,3,B0           ; |83|
           MVK     .S1     0x4000,A4
   [ B1]   MVK     .S2     0x2,B1            ; init prolog collapse predicate
           ; BRANCH OCCURS                   ; |83|
;** --------------------------------------------------------------------------*
   [ B0]   B       .S1     L8                ; |83|
           SUBAW   .D1     A0,4,A7
           CMPEQ   .L2     B5,4,B1           ; |83|
           MVK     .S1     0x4000,A6
   [ B0]   MVK     .S2     0x3,B0            ; init prolog collapse predicate
           NOP             1
           ; BRANCH OCCURS                   ; |83|
;** --------------------------------------------------------------------------*
   [!B1]   B       .S1     L20               ; |83|
           SUBAW   .D1     A0,4,A9
           MVK     .S1     0x4000,A7

   [!B1]   MV      .L1X    B12,A0            ; |100|
|| [!B1]   ZERO    .S1     A6                ; |109|

   [!B1]   STW     .D1T1   A6,*+A0(24)       ; |109|
|| [!B1]   MV      .L1X    B13,A5            ; |109|
```

```
   [!B1]   LDH     .D1T1   *A5,A0           ; |101|
           ; BRANCH OCCURS                 ; |83|
;** --------------------------------------------------------------------------*
           MVK     .S2     0x1,B1          ; init prolog collapse predicate

           MV      .L2X    A9,B5
||         MV      .L1X    B4,A1
||         MVK     .S2     0x2,B0          ; init prolog collapse predicate
||         MVK     .S1     0x4000,A2       ; init prolog collapse predicate

;** --------------------------------------------------------------------------*
L5:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L6:    ; PIPED LOOP KERNEL

           ADD     .L1     A7,A3,A4        ; |83|
|| [!A2]   STH     .D2T2   B7,*-B5(10)     ; |83|
||         SHR     .S2X    A0,15,B7        ; @|83|
||         SHR     .S1     A6,15,A0        ; @@|83|
||         LDH     .D1T1   *++A9(16),A3    ; @@@|83|

           SHR     .S1     A4,15,A4        ; |83|
|| [!A2]   LDH     .D1T1   *-A9(38),A5     ; |83|
|| [!B0]   STH     .D2T2   B7,*+B5(2)      ; @|83|
||         ADD     .L2X    A7,B4,B4        ; @|83|

           SHL     .S1     A8,11,A6        ; |83|
|| [!A2]   STH     .D1T1   A4,*-A9(34)     ; |83|
|| [!B0]   LDH     .D2T2   *+B5(4),B8      ; @|83|
||         SHR     .S2     B4,15,B4        ; @|83|

           SHL     .S2     B6,11,B6        ; @|83|
|| [!B1]   STH     .D2T1   A0,*++B5(16)    ; @@|83|

   [ A1]   SUB     .L1     A1,1,A1         ; |83|
||         ADD     .L2X    A7,B6,B6        ; @|83|
|| [!B0]   LDH     .D1T1   *-A9(18),A4     ; @|83|
|| [!B1]   LDH     .D2T2   *+B5(2),B9      ; @@|83|

   [ A1]   B       .S1     L6              ; |83|
||         SHR     .S2     B6,15,B7        ; @|83|
|| [!B1]   LDH     .D1T1   *-A9(8),A0      ; @@|83|

           SHL     .S2X    A5,11,B4        ; |83|
||         ADD     .L1     A7,A6,A5        ; |83|
|| [!B0]   STH     .D1T2   B4,*-A9(24)     ; @|83|

           ADD     .L2X    A7,B4,B6        ; |83|
||         SHL     .S2     B8,11,B4        ; @|83|
||         SHL     .S1     A3,11,A3        ; @@@|83|
```

```
           SHR     .S2     B6,15,B4          ; |83|
||         SHR     .S1     A5,15,A5          ; |83|
|| [!B0]   LDH     .D1T1   *-A9(20),A8       ; @|83|
||         ADD     .L2X    A7,B4,B8          ; @|83|
|| [!B1]   LDH     .D2T2   *+B5(6),B6        ; @@|83|
||         ADD     .L1     A7,A3,A6          ; @@@|83|

   [ B1]   SUB     .L2     B1,1,B1           ;
|| [ A2]   MPYSU   .M1     2,A2,A2           ;
|| [!A2]   STH     .D1T2   B4,*-A9(38)       ; |83|
||         SHR     .S2     B8,15,B4          ; @|83|
||         SHL     .S1X    B9,11,A3          ; @@|83|

   [ B0]   SUB     .L2     B0,1,B0           ;
|| [!A2]   STH     .D1T1   A5,*-A9(36)       ; |83|
|| [!B0]   STH     .D2T2   B4,*-B5(12)       ; @|83|
||         SHL     .S1     A4,11,A3          ; @|83|
||         SHL     .S2X    A0,11,B4          ; @@|83|
||         ADD     .L1     A7,A3,A0          ; @@|83|

;** --------------------------------------------------------------------------*
L7:    ; PIPED LOOP EPILOG

           ADD     .L1     A7,A3,A4          ; @|83|
||         STH     .D2T2   B7,*-B5(10)       ; @|83|
||         SHR     .S2X    A0,15,B7          ; @@|83|
||         SHR     .S1     A6,15,A0          ; @@@|83|

           SHR     .S1     A4,15,A4          ; @|83|
||         LDH     .D1T1   *-A9(22),A5       ; @|83|
||         STH     .D2T2   B7,*+B5(2)        ; @@|83|
||         ADD     .L2X    A7,B4,B4          ; @@|83|

           SHL     .S1     A8,11,A6          ; @|83|
||         STH     .D1T1   A4,*-A9(18)       ; @|83|
||         LDH     .D2T2   *+B5(4),B8        ; @@|83|
||         SHR     .S2     B4,15,B4          ; @@|83|

           SHL     .S2     B6,11,B6          ; @@|83|
||         STH     .D2T1   A0,*++B5(16)      ; @@@|83|

           ADD     .L2X    A7,B6,B6          ; @@|83|
||         LDH     .D1T1   *-A9(2),A4        ; @@|83|
||         LDH     .D2T2   *+B5(2),B9        ; @@@|83|

           SHR     .S2     B6,15,B7          ; @@|83|
||         LDH     .D1T1   *+A9(8),A0        ; @@@|83|

           SHL     .S2X    A5,11,B4          ; @|83|
||         ADD     .L1     A7,A6,A5          ; @|83|
||         STH     .D1T2   B4,*-A9(8)        ; @@|83|
```

```
           ADD     .L2X    A7,B4,B6          ; @|83|
||         SHL     .S2     B8,11,B4          ; @@|83|

           SHR     .S2     B6,15,B4          ; @|83|
||         SHR     .S1     A5,15,A5          ; @|83|
||         LDH     .D1T1   *-A9(4),A8        ; @@|83|
||         ADD     .L2X    A7,B4,B8          ; @@|83|
||         LDH     .D2T2   *+B5(6),B6        ; @@@|83|

           STH     .D1T2   B4,*-A9(22)       ; @|83|
||         SHR     .S2     B8,15,B4          ; @@|83|
||         SHL     .S1X    B9,11,A3          ; @@@|83|

           STH     .D1T1   A5,*-A9(20)       ; @|83|
||         STH     .D2T2   B4,*-B5(12)       ; @@|83|
||         SHL     .S1     A4,11,A3          ; @@|83|
||         SHL     .S2X    A0,11,B4          ; @@@|83|
||         ADD     .L1     A7,A3,A0          ; @@@|83|

           ADD     .L1     A7,A3,A4          ; @@|83|
||         STH     .D2T2   B7,*-B5(10)       ; @@|83|
||         SHR     .S2X    A0,15,B7          ; @@@|83|

           SHR     .S1     A4,15,A4          ; @@|83|
||         LDH     .D1T1   *-A9(6),A5        ; @@|83|
||         STH     .D2T2   B7,*+B5(2)        ; @@@|83|
||         ADD     .L2X    A7,B4,B4          ; @@@|83|

           SHL     .S1     A8,11,A6          ; @@|83|
||         STH     .D1T1   A4,*-A9(2)        ; @@|83|
||         LDH     .D2T2   *+B5(4),B8        ; @@@|83|
||         SHR     .S2     B4,15,B4          ; @@@|83|

           SHL     .S2     B6,11,B6          ; @@@|83|

           ADD     .L2X    A7,B6,B6          ; @@@|83|
||         LDH     .D1T1   *+A9(14),A4       ; @@@|83|

           SHR     .S2     B6,15,B7          ; @@@|83|

           SHL     .S2X    A5,11,B4          ; @@|83|
||         ADD     .L1     A7,A6,A5          ; @@|83|
||         STH     .D1T2   B4,*+A9(8)        ; @@@|83|

           ADD     .L2X    A7,B4,B6          ; @@|83|
||         SHL     .S2     B8,11,B4          ; @@@|83|

           SHR     .S2     B6,15,B4          ; @@|83|
||         SHR     .S1     A5,15,A5          ; @@|83|
||         LDH     .D1T1   *+A9(12),A8       ; @@@|83|
```

```
||         ADD     .L2X    A7,B4,B8          ; @@@|83|

           STH     .D1T2   B4,*-A9(6)        ; @@|83|
||         SHR     .S2     B8,15,B4          ; @@@|83|

           STH     .D1T1   A5,*-A9(4)        ; @@|83|
||         STH     .D2T2   B4,*+B5(4)        ; @@@|83|
||         SHL     .S1     A4,11,A3          ; @@@|83|

           ADD     .L1     A7,A3,A4          ; @@@|83|
||         STH     .D2T2   B7,*+B5(6)        ; @@@|83|

           SHR     .S1     A4,15,A4          ; @@@|83|
||         LDH     .D1T1   *+A9(10),A5       ; @@@|83|

           SHL     .S1     A8,11,A6          ; @@@|83|
||         STH     .D1T1   A4,*+A9(14)       ; @@@|83|

           NOP             3

           SHL     .S2X    A5,11,B4          ; @@@|83|
||         ADD     .L1     A7,A6,A5          ; @@@|83|

           ADD     .L2X    A7,B4,B6          ; @@@|83|

           SHR     .S2     B6,15,B4          ; @@@|83|
||         SHR     .S1     A5,15,A5          ; @@@|83|

           STH     .D1T2   B4,*+A9(10)       ; @@@|83|
           STH     .D1T1   A5,*+A9(12)       ; @@@|83|
;** --------------------------------------------------------------------------*
           B       .S1     L22               ; |83|

           MV      .L1X    B12,A0            ; |100|
||         ZERO    .S1     A6                ; |109|

           STW     .D1T1   A6,*+A0(24)       ; |109|
||         MV      .L1X    B13,A5            ; |109|

           LDH     .D1T1   *A5,A0            ; |101|
           LDH     .D1T1   *A5,A3            ; |111|
           NOP             1
           ; BRANCH OCCURS                   ; |83|
;** --------------------------------------------------------------------------*
L8:

           MVK     .S2     0x2,B1            ; init prolog collapse predicate
||         ZERO    .L2     B2

           MVKH    .S2     0x10000,B2        ; init prolog collapse predicate
||         MV      .L1X    B4,A1
```

```
||         MV      .L2X    A7,B7
||         MVK     .S1     0x4,A2           ; init prolog collapse predicate

;** --------------------------------------------------------------------------*
L9:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L10:    ; PIPED LOOP KERNEL

           ADD     .L2X    A6,B4,B4         ; @|82|
|| [!B1]   LDH     .D2T2   *+B7(6),B5       ; @@|82|

           SHR     .S2     B4,15,B4         ; @|82|

           SHL     .S1     A4,12,A0         ; |82|
|| [!B0]   STH     .D1T2   B4,*-A7(24)      ; @|82|
|| [!B2]   STH     .D2T1   A0,*++B7(16)     ; @@@|82|

           ADD     .L1     A6,A0,A0         ; |82|
|| [!B0]   LDH     .D1T1   *-A7(22),A4      ; @|82|
|| [!B2]   LDH     .D2T2   *+B7(2),B6       ; @@@|82|

           SHR     .S1     A0,15,A0         ; |82|
|| [!B2]   LDH     .D2T2   *+B7(4),B4       ; @@@|82|

   [!A2]   STH     .D1T1   A0,*-A7(36)      ; |82|
||         SHL     .S2     B5,12,B5         ; @@|82|

   [!A2]   LDH     .D1T1   *-A7(34),A3      ; |82|
||         ADD     .L2X    A6,B5,B5         ; @@|82|

           SHR     .S2     B5,15,B5         ; @@|82|
||         LDH     .D1T1   *++A7(16),A0     ; @@@@|82|

   [ A1]   SUB     .L1     A1,1,A1          ; |82|
||         SHL     .S2X    A4,12,B5         ; @|82|
|| [!B1]   STH     .D2T2   B5,*-B7(10)      ; @@|82|
||         SHL     .S1X    B6,12,A4         ; @@@|82|

   [ A1]   B       .S1     L10              ; |82|
||         ADD     .L2X    A6,B5,B5         ; @|82|
|| [!B1]   LDH     .D1T1   *-A7(24),A5      ; @@|82|
||         SHL     .S2     B4,12,B4         ; @@@|82|

           SHR     .S2     B5,15,B5         ; @|82|
||         ADD     .L1     A6,A4,A4         ; @@@|82|

           SHL     .S1     A3,12,A3         ; |82|
|| [!B0]   STH     .D1T2   B5,*-A7(38)      ; @|82|
||         SHR     .S2X    A4,15,B5         ; @@@|82|

           ADD     .L1     A6,A3,A0         ; |82|
```

```
|| [!B0]   LDH     .D1T1   *-A7(36),A4       ; @|82|
|| [!B2]   STH     .D2T2   B5,*+B7(2)        ; @@@|82|
||         ADD     .L2X    A6,B4,B4          ; @@@|82|
||         SHL     .S1     A0,12,A3          ; @@@@|82|

   [ B1]   SUB     .L2     B1,1,B1           ;
||         SHR     .S1     A0,15,A3          ; |82|
||         SHR     .S2     B4,15,B4          ; @@@|82|
||         ADD     .L1     A6,A3,A0          ; @@@@|82|

   [ B2]   MPYSU   .M2     2,B2,B2           ;
|| [ B0]   SUB     .L2     B0,1,B0           ;
|| [ A2]   SUB     .L1     A2,1,A2           ;
|| [!A2]   STH     .D1T1   A3,*-A7(50)       ; |82|
||         SHL     .S2X    A5,12,B4          ; @@|82|
|| [!B2]   STH     .D2T2   B4,*+B7(4)        ; @@@|82|
||         SHR     .S1     A0,15,A0          ; @@@@|82|

;** --------------------------------------------------------------------------*
L11:    ; PIPED LOOP EPILOG

           ADD     .L2X    A6,B4,B4          ; @@|82|
||         LDH     .D2T2   *+B7(6),B5        ; @@@|82|

           SHR     .S2     B4,15,B4          ; @@|82|

           SHL     .S1     A4,12,A0          ; @|82|
||         STH     .D1T2   B4,*-A7(24)       ; @@|82|
||         STH     .D2T1   A0,*++B7(16)      ; @@@@|82|

           ADD     .L1     A6,A0,A0          ; @|82|
||         LDH     .D1T1   *-A7(22),A4       ; @@|82|
||         LDH     .D2T2   *+B7(2),B6        ; @@@@|82|

           SHR     .S1     A0,15,A0          ; @|82|
||         LDH     .D2T2   *+B7(4),B4        ; @@@@|82|

           STH     .D1T1   A0,*-A7(36)       ; @|82|
||         SHL     .S2     B5,12,B5          ; @@@|82|

           LDH     .D1T1   *-A7(34),A3       ; @|82|
||         ADD     .L2X    A6,B5,B5          ; @@@|82|

           SHR     .S2     B5,15,B5          ; @@@|82|

           SHL     .S2X    A4,12,B5          ; @@|82|
||         STH     .D2T2   B5,*-B7(10)       ; @@@|82|
||         SHL     .S1X    B6,12,A4          ; @@@@|82|

           ADD     .L2X    A6,B5,B5          ; @@|82|
||         LDH     .D1T1   *-A7(8),A5        ; @@@|82|
```

```
||          SHL     .S2     B4,12,B4          ; @@@@|82|

            SHR     .S2     B5,15,B5          ; @@|82|
||          ADD     .L1     A6,A4,A4          ; @@@@|82|

            SHL     .S1     A3,12,A3          ; @|82|
||          STH     .D1T2   B5,*-A7(22)       ; @@|82|
||          SHR     .S2X    A4,15,B5          ; @@@@|82|

            ADD     .L1     A6,A3,A0          ; @|82|
||          LDH     .D1T1   *-A7(20),A4       ; @@|82|
||          STH     .D2T2   B5,*+B7(2)        ; @@@@|82|
||          ADD     .L2X    A6,B4,B4          ; @@@@|82|

            SHR     .S1     A0,15,A3          ; @|82|
||          SHR     .S2     B4,15,B4          ; @@@@|82|

            STH     .D1T1   A3,*-A7(34)       ; @|82|
||          SHL     .S2X    A5,12,B4          ; @@@|82|
||          STH     .D2T2   B4,*+B7(4)        ; @@@@|82|

            ADD     .L2X    A6,B4,B4          ; @@@|82|
||          LDH     .D2T2   *+B7(6),B5        ; @@@@|82|

            SHR     .S2     B4,15,B4          ; @@@|82|

            SHL     .S1     A4,12,A0          ; @@|82|
||          STH     .D1T2   B4,*-A7(8)        ; @@@|82|

            ADD     .L1     A6,A0,A0          ; @@|82|
||          LDH     .D1T1   *-A7(6),A4        ; @@@|82|

            SHR     .S1     A0,15,A0          ; @@|82|

            STH     .D1T1   A0,*-A7(20)       ; @@|82|
||          SHL     .S2     B5,12,B5          ; @@@@|82|

            LDH     .D1T1   *-A7(18),A3       ; @@|82|
||          ADD     .L2X    A6,B5,B5          ; @@@@|82|

            SHR     .S2     B5,15,B5          ; @@@@|82|

            SHL     .S2X    A4,12,B5          ; @@@|82|
||          STH     .D2T2   B5,*+B7(6)        ; @@@@|82|

            ADD     .L2X    A6,B5,B5          ; @@@|82|
||          LDH     .D1T1   *+A7(8),A5        ; @@@@|82|

            SHR     .S2     B5,15,B5          ; @@@|82|

            SHL     .S1     A3,12,A3          ; @@|82|
```

```
||         STH     .D1T2   B5,*-A7(6)        ; @@@|82|

           ADD     .L1     A6,A3,A0          ; @@|82|
||         LDH     .D1T1   *-A7(4),A4        ; @@@|82|

           SHR     .S1     A0,15,A3          ; @@|82|

           STH     .D1T1   A3,*-A7(18)       ; @@|82|
||         SHL     .S2X    A5,12,B4          ; @@@@|82|

           ADD     .L2X    A6,B4,B4          ; @@@@|82|
           SHR     .S2     B4,15,B4          ; @@@@|82|

           SHL     .S1     A4,12,A0          ; @@@|82|
||         STH     .D1T2   B4,*+A7(8)        ; @@@@|82|

           ADD     .L1     A6,A0,A0          ; @@@|82|
||         LDH     .D1T1   *+A7(10),A4       ; @@@@|82|

           SHR     .S1     A0,15,A0          ; @@@|82|
           STH     .D1T1   A0,*-A7(4)        ; @@@|82|
           LDH     .D1T1   *-A7(2),A3        ; @@@|82|
           NOP             1
           SHL     .S2X    A4,12,B5          ; @@@@|82|
           ADD     .L2X    A6,B5,B5          ; @@@@|82|
           SHR     .S2     B5,15,B5          ; @@@@|82|

           SHL     .S1     A3,12,A3          ; @@@|82|
||         STH     .D1T2   B5,*+A7(10)       ; @@@@|82|

           ADD     .L1     A6,A3,A0          ; @@@|82|
||         LDH     .D1T1   *+A7(12),A4       ; @@@@|82|

           SHR     .S1     A0,15,A3          ; @@@|82|
           STH     .D1T1   A3,*-A7(2)        ; @@@|82|
           NOP             2
           SHL     .S1     A4,12,A0          ; @@@@|82|
           ADD     .L1     A6,A0,A0          ; @@@@|82|
           SHR     .S1     A0,15,A0          ; @@@@|82|
           STH     .D1T1   A0,*+A7(12)       ; @@@@|82|
           LDH     .D1T1   *+A7(14),A3       ; @@@@|82|
           NOP             4
           SHL     .S1     A3,12,A3          ; @@@@|82|
           ADD     .L1     A6,A3,A0          ; @@@@|82|
           SHR     .S1     A0,15,A3          ; @@@@|82|
           STH     .D1T1   A3,*+A7(14)       ; @@@@|82|
;** --------------------------------------------------------------------------*
           B       .S1     L22               ; |82|

           MV      .L1X    B12,A0            ; |100|
||         ZERO    .S1     A6                ; |109|
```

```
           STW     .D1T1   A6,*+A0(24)       ; |109|
||         MV      .L1X    B13,A5            ; |109|

           LDH     .D1T1   *A5,A0            ; |101|
           LDH     .D1T1   *A5,A3            ; |111|
           NOP             1
           ; BRANCH OCCURS                   ; |82|
;** --------------------------------------------------------------------------*
L12:
           MVK     .S2     0x1,B2            ; init prolog collapse predicate

           MV      .L2X    A5,B5
||         MV      .L1X    B4,A1
||         MVK     .S2     0x3,B0            ; init prolog collapse predicate
||         MVK     .S1     0x4,A2            ; init prolog collapse predicate

;** --------------------------------------------------------------------------*
L13:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L14:    ; PIPED LOOP KERNEL

           ADD     .L1     A4,A3,A3          ; |81|
|| [!A2]   STH     .D1T2   B4,*-A5(40)       ; |81|
||         SHL     .S2     B9,13,B4          ; @|81|

   [!A2]   STH     .D1T2   B6,*-A5(38)       ; |81|
||         SHR     .S1     A3,15,A3          ; |81|
||         SHR     .S2     B8,15,B6          ; @|81|
||         ADD     .L2X    A4,B4,B4          ; @|81|

   [!A2]   STH     .D1T1   A3,*-A5(36)       ; |81|
|| [!B0]   STH     .D2T2   B6,*-B5(10)       ; @|81|
||         SHR     .S2     B4,15,B4          ; @|81|

   [!A2]   LDH     .D1T1   *-A5(34),A7       ; |81|
||         SHL     .S2X    A6,13,B4          ; @|81|
|| [!B0]   STH     .D2T2   B4,*-B5(12)       ; @|81|
||         SHL     .S1X    B7,13,A3          ; @@|81|

           ADD     .L2X    A4,B4,B6          ; @|81|
|| [!B0]   LDH     .D1T1   *-A5(24),A6       ; @|81|
||         ADD     .L1     A4,A3,A0          ; @@|81|
||         SHL     .S1     A0,13,A3          ; @@@|81|

   [ A1]   SUB     .S1     A1,1,A1           ; |81|
|| [!B0]   LDH     .D1T1   *-A5(20),A3       ; @|81|
|| [!B1]   LDH     .D2T2   *+B5(6),B8        ; @@|81|
||         SHR     .S2X    A0,15,B4          ; @@|81|
||         ADD     .L1     A4,A3,A0          ; @@@|81|
```

```
   [ A1]   B       .S1     L14             ; |81|
||         SHR     .S2     B6,15,B6        ; @|81|
|| [!B1]   STH     .D2T2   B4,*+B5(2)      ; @@|81|

   [!B1]   LDH     .D2T2   *+B5(4),B9      ; @@|81|
||         SHR     .S1     A0,15,A0        ; @@@|81|

           SHL     .S1     A7,13,A0        ; |81|
|| [!B2]   STH     .D2T1   A0,*++B5(16)    ; @@@|81|

   [ B2]   SUB     .L2     B2,1,B2         ;
||         ADD     .L1     A4,A0,A0        ; |81|
||         SHL     .S2X    A6,13,B4        ; @|81|
|| [!B1]   LDH     .D1T1   *-A5(6),A6      ; @@|81|
|| [!B2]   LDH     .D2T2   *+B5(2),B7      ; @@@|81|

   [ B1]   SUB     .D2     B1,1,B1         ;
||         SHR     .S1     A0,15,A7        ; |81|
||         ADD     .L2X    A4,B4,B8        ; @|81|
||         SHL     .S2     B8,13,B4        ; @@|81|
||         LDH     .D1T1   *++A5(16),A0    ; @@@@|81|

   [ B0]   SUB     .D2     B0,1,B0         ;
|| [ A2]   SUB     .L1     A2,1,A2         ;
|| [!A2]   STH     .D1T1   A7,*-A5(50)     ; |81|
||         SHL     .S1     A3,13,A3        ; @|81|
||         SHR     .S2     B8,15,B4        ; @|81|
||         ADD     .L2X    A4,B4,B8        ; @@|81|

;** --------------------------------------------------------------------------*
L15:    ; PIPED LOOP EPILOG

           ADD     .L1     A4,A3,A3        ; @|81|
||         STH     .D1T2   B4,*-A5(40)     ; @|81|
||         SHL     .S2     B9,13,B4        ; @@|81|

           STH     .D1T2   B6,*-A5(38)     ; @|81|
||         SHR     .S1     A3,15,A3        ; @|81|
||         SHR     .S2     B8,15,B6        ; @@|81|
||         ADD     .L2X    A4,B4,B4        ; @@|81|

           STH     .D1T1   A3,*-A5(36)     ; @|81|
||         STH     .D2T2   B6,*-B5(10)     ; @@|81|
||         SHR     .S2     B4,15,B4        ; @@|81|

           LDH     .D1T1   *-A5(34),A7     ; @|81|
||         SHL     .S2X    A6,13,B4        ; @@|81|
||         STH     .D2T2   B4,*-B5(12)     ; @@|81|
||         SHL     .S1X    B7,13,A3        ; @@@|81|

           ADD     .L2X    A4,B4,B6        ; @@|81|
```

```
||          LDH     .D1T1   *-A5(24),A6       ; @@|81|
||          ADD     .L1     A4,A3,A0          ; @@@|81|
||          SHL     .S1     A0,13,A3          ; @@@@|81|

            LDH     .D1T1   *-A5(20),A3       ; @@|81|
||          LDH     .D2T2   *+B5(6),B8        ; @@@|81|
||          SHR     .S2X    A0,15,B4          ; @@@|81|
||          ADD     .L1     A4,A3,A0          ; @@@@|81|

            SHR     .S2     B6,15,B6          ; @@|81|
||          STH     .D2T2   B4,*+B5(2)        ; @@@|81|

            LDH     .D2T2   *+B5(4),B9        ; @@@|81|
||          SHR     .S1     A0,15,A0          ; @@@@|81|

            SHL     .S1     A7,13,A0          ; @|81|
||          STH     .D2T1   A0,*++B5(16)      ; @@@@|81|

            ADD     .L1     A4,A0,A0          ; @|81|
||          SHL     .S2X    A6,13,B4          ; @@|81|
||          LDH     .D1T1   *-A5(6),A6        ; @@@|81|
||          LDH     .D2T2   *+B5(2),B7        ; @@@@|81|

            SHR     .S1     A0,15,A7          ; @|81|
||          ADD     .L2X    A4,B4,B8          ; @@|81|
||          SHL     .S2     B8,13,B4          ; @@@|81|

            STH     .D1T1   A7,*-A5(34)       ; @|81|
||          SHL     .S1     A3,13,A3          ; @@|81|
||          SHR     .S2     B8,15,B4          ; @@|81|
||          ADD     .L2X    A4,B4,B8          ; @@@|81|

            ADD     .L1     A4,A3,A3          ; @@|81|
||          STH     .D1T2   B4,*-A5(24)       ; @@|81|
||          SHL     .S2     B9,13,B4          ; @@@|81|

            STH     .D1T2   B6,*-A5(22)       ; @@|81|
||          SHR     .S1     A3,15,A3          ; @@|81|
||          SHR     .S2     B8,15,B6          ; @@@|81|
||          ADD     .L2X    A4,B4,B4          ; @@@|81|

            STH     .D1T1   A3,*-A5(20)       ; @@|81|
||          STH     .D2T2   B6,*-B5(10)       ; @@@|81|
||          SHR     .S2     B4,15,B4          ; @@@|81|

            LDH     .D1T1   *-A5(18),A7       ; @@|81|
||          SHL     .S2X    A6,13,B4          ; @@@|81|
||          STH     .D2T2   B4,*-B5(12)       ; @@@|81|
||          SHL     .S1X    B7,13,A3          ; @@@@|81|

            ADD     .L2X    A4,B4,B6          ; @@@|81|
```

```
||          LDH     .D1T1   *-A5(8),A6        ; @@@|81|
||          ADD     .L1     A4,A3,A0          ; @@@@|81|

            LDH     .D1T1   *-A5(4),A3        ; @@@|81|
||          LDH     .D2T2   *+B5(6),B8        ; @@@@|81|
||          SHR     .S2X    A0,15,B4          ; @@@@|81|

            SHR     .S2     B6,15,B6          ; @@@|81|
||          STH     .D2T2   B4,*+B5(2)        ; @@@@|81|

            LDH     .D2T2   *+B5(4),B9        ; @@@@|81|
            SHL     .S1     A7,13,A0          ; @@|81|

            ADD     .L1     A4,A0,A0          ; @@|81|
||          SHL     .S2X    A6,13,B4          ; @@@|81|
||          LDH     .D1T1   *+A5(10),A6       ; @@@@|81|

            SHR     .S1     A0,15,A7          ; @@|81|
||          ADD     .L2X    A4,B4,B8          ; @@@|81|
||          SHL     .S2     B8,13,B4          ; @@@@|81|

            STH     .D1T1   A7,*-A5(18)       ; @@|81|
||          SHL     .S1     A3,13,A3          ; @@@|81|
||          SHR     .S2     B8,15,B4          ; @@@|81|
||          ADD     .L2X    A4,B4,B8          ; @@@@|81|

            ADD     .L1     A4,A3,A3          ; @@@|81|
||          STH     .D1T2   B4,*-A5(8)        ; @@@|81|
||          SHL     .S2     B9,13,B4          ; @@@@|81|

            STH     .D1T2   B6,*-A5(6)        ; @@@|81|
||          SHR     .S1     A3,15,A3          ; @@@|81|
||          SHR     .S2     B8,15,B6          ; @@@@|81|
||          ADD     .L2X    A4,B4,B4          ; @@@@|81|

            STH     .D1T1   A3,*-A5(4)        ; @@@|81|
||          STH     .D2T2   B6,*+B5(6)        ; @@@@|81|
||          SHR     .S2     B4,15,B4          ; @@@@|81|

            LDH     .D1T1   *-A5(2),A7        ; @@@|81|
||          SHL     .S2X    A6,13,B4          ; @@@@|81|
||          STH     .D2T2   B4,*+B5(4)        ; @@@@|81|

            ADD     .L2X    A4,B4,B6          ; @@@@|81|
||          LDH     .D1T1   *+A5(8),A6        ; @@@@|81|

            LDH     .D1T1   *+A5(12),A3       ; @@@@|81|
            SHR     .S2     B6,15,B6          ; @@@@|81|
            NOP             1
            SHL     .S1     A7,13,A0          ; @@@|81|
```

```
           ADD     .L1     A4,A0,A0          ; @@@|81|
||         SHL     .S2X    A6,13,B4          ; @@@@|81|

           SHR     .S1     A0,15,A7          ; @@@|81|
||         ADD     .L2X    A4,B4,B8          ; @@@@|81|

           STH     .D1T1   A7,*-A5(2)        ; @@@|81|
||         SHL     .S1     A3,13,A3          ; @@@@|81|
||         SHR     .S2     B8,15,B4          ; @@@@|81|

           ADD     .L1     A4,A3,A3          ; @@@@|81|
||         STH     .D1T2   B4,*+A5(8)        ; @@@@|81|

           STH     .D1T2   B6,*+A5(10)       ; @@@@|81|
||         SHR     .S1     A3,15,A3          ; @@@@|81|

           STH     .D1T1   A3,*+A5(12)       ; @@@@|81|
           LDH     .D1T1   *+A5(14),A7       ; @@@@|81|
           NOP             4
           SHL     .S1     A7,13,A0          ; @@@@|81|
           ADD     .L1     A4,A0,A0          ; @@@@|81|
           SHR     .S1     A0,15,A7          ; @@@@|81|
           STH     .D1T1   A7,*+A5(14)       ; @@@@|81|
;** --------------------------------------------------------------------------*
           B       .S1     L22               ; |81|

           MV      .L1X    B12,A0            ; |100|
||         ZERO    .S1     A6                ; |109|

           STW     .D1T1   A6,*+A0(24)       ; |109|
||         MV      .L1X    B13,A5            ; |109|

           LDH     .D1T1   *A5,A0            ; |101|
           LDH     .D1T1   *A5,A3            ; |111|
           NOP             1
           ; BRANCH OCCURS                   ; |81|
;** --------------------------------------------------------------------------*
L16:

           MV      .L2X    A4,B7
||         MVK     .S1     0x2,A1            ; init prolog collapse predicate

;** --------------------------------------------------------------------------*
L17:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L18:    ; PIPED LOOP KERNEL

           ADDK    .S2     16384,B4          ; |80|
||         SHL     .S1     A6,14,A5          ; |80|
||         LDH     .D1T1   *++A4(16),A6      ; @@|80|
```

```
   [!A1]   LDH     .D1T1   *-A4(18),A3       ; |80|
||         SHR     .S2     B4,15,B4          ; |80|
||         ADDK    .S1     16384,A5          ; |80|

           SHR     .S2     B6,15,B4          ; |80|
|| [!A1]   STH     .D1T2   B4,*-A4(28)       ; |80|
||         SHR     .S1     A5,15,A5          ; |80|

           SHL     .S2X    A0,14,B4          ; |80|
|| [!A1]   STH     .D1T2   B4,*-A4(24)       ; |80|
||         SHL     .S1X    B5,14,A0          ; @|80|

           ADDK    .S2     16384,B4          ; |80|
|| [!A1]   STH     .D1T1   A5,*-A4(26)       ; |80|
||         ADDK    .S1     16384,A0          ; @|80|

   [!A1]   LDH     .D1T1   *-A4(22),A0       ; |80|
||         SHR     .S1     A0,15,A5          ; @|80|
||         SHL     .S2X    A6,14,B5          ; @@|80|

           SHL     .S1     A3,14,A5          ; |80|
|| [!A2]   STH     .D2T1   A5,*+B7(2)        ; @|80|
||         ADDK    .S2     16384,B5          ; @@|80|

           ADDK    .S1     16384,A5          ; |80|
|| [!A2]   LDH     .D1T1   *-A4(8),A3        ; @|80|
||         SHR     .S2     B5,15,B5          ; @@|80|

   [ B0]   SUB     .L2     B0,1,B0           ; |80|
||         SHR     .S2     B4,15,B4          ; |80|
||         SHR     .S1     A5,15,A7          ; |80|

   [ B0]   B       .S2     L18               ; |80|
|| [!A2]   LDH     .D1T1   *-A4(12),A5       ; @|80|

           SHL     .S1     A0,14,A0          ; |80|
|| [!A2]   LDH     .D1T1   *-A4(10),A6       ; @|80|
||         STH     .D2T2   B5,*++B7(16)      ; @@|80|

   [!A1]   STH     .D1T1   A7,*-A4(18)       ; |80|
||         ADDK    .S1     16384,A0          ; |80|
||         LDH     .D2T2   *+B7(2),B5        ; @@|80|

           SHR     .S1     A0,15,A3          ; |80|
|| [!A2]   LDH     .D1T1   *-A4(4),A0        ; @|80|
||         SHL     .S2X    A3,14,B6          ; @|80|

   [!A1]   STH     .D1T1   A3,*-A4(22)       ; |80|
||         ADDK    .S2     16384,B6          ; @|80|

   [ A2]   SUB     .S1     A2,1,A2           ;
```

```
|| [ A1]   SUB     .L1     A1,1,A1          ;
|| [!A1]   STH     .D1T2   B4,*-A4(20)      ; |80|
||         SHL     .S2X    A5,14,B4         ; @|80|

;** --------------------------------------------------------------------------*
L19:    ; PIPED LOOP EPILOG

           ADDK    .S2     16384,B4         ; @|80|
||         SHL     .S1     A6,14,A5         ; @|80|

           LDH     .D1T1   *-A4(2),A3       ; @|80|
||         SHR     .S2     B4,15,B4         ; @|80|
||         ADDK    .S1     16384,A5         ; @|80|

           SHR     .S2     B6,15,B4         ; @|80|
||         STH     .D1T2   B4,*-A4(12)      ; @|80|
||         SHR     .S1     A5,15,A5         ; @|80|

           SHL     .S2X    A0,14,B4         ; @|80|
||         STH     .D1T2   B4,*-A4(8)       ; @|80|
||         SHL     .S1X    B5,14,A0         ; @@|80|

           ADDK    .S2     16384,B4         ; @|80|
||         STH     .D1T1   A5,*-A4(10)      ; @|80|
||         ADDK    .S1     16384,A0         ; @@|80|

           LDH     .D1T1   *-A4(6),A0       ; @|80|
||         SHR     .S1     A0,15,A5         ; @@|80|

           SHL     .S1     A3,14,A5         ; @|80|
||         STH     .D2T1   A5,*+B7(2)       ; @@|80|

           ADDK    .S1     16384,A5         ; @|80|
||         LDH     .D1T1   *+A4(8),A3       ; @@|80|

           SHR     .S2     B4,15,B4         ; @|80|
||         SHR     .S1     A5,15,A7         ; @|80|

           LDH     .D1T1   *+A4(4),A5       ; @@|80|

           SHL     .S1     A0,14,A0         ; @|80|
||         LDH     .D1T1   *+A4(6),A6       ; @@|80|

           STH     .D1T1   A7,*-A4(2)       ; @|80|
||         ADDK    .S1     16384,A0         ; @|80|

           SHR     .S1     A0,15,A3         ; @|80|
||         LDH     .D1T1   *+A4(12),A0      ; @@|80|
||         SHL     .S2X    A3,14,B6         ; @@|80|

           STH     .D1T1   A3,*-A4(6)       ; @|80|
```

```
||         ADDK    .S2     16384,B6          ; @@|80|

           STH     .D1T2   B4,*-A4(4)        ; @|80|
||         SHL     .S2X    A5,14,B4          ; @@|80|

           ADDK    .S2     16384,B4          ; @@|80|
||         SHL     .S1     A6,14,A5          ; @@|80|

           LDH     .D1T1   *++A4(14),A3      ; @@|80|
||         SHR     .S2     B4,15,B4          ; @@|80|
||         ADDK    .S1     16384,A5          ; @@|80|

           SHR     .S2     B6,15,B4          ; @@|80|
||         STH     .D1T2   B4,*++A4(4)       ; @@|80|
||         SHR     .S1     A5,15,A5          ; @@|80|

           SHL     .S2X    A0,14,B4          ; @@|80|
||         STH     .D1T2   B4,*++A4(8)       ; @@|80|

           ADDK    .S2     16384,B4          ; @@|80|
||         STH     .D1T1   A5,*++A4(6)       ; @@|80|

           LDH     .D1T1   *++A4(10),A0      ; @@|80|
           SHL     .S1     A3,14,A5          ; @@|80|
           ADDK    .S1     16384,A5          ; @@|80|

           SHR     .S2     B4,15,B4          ; @@|80|
||         SHR     .S1     A5,15,A7          ; @@|80|

           NOP             1
           SHL     .S1     A0,14,A0          ; @@|80|

           STH     .D1T1   A7,*++A4(14)      ; @@|80|
||         ADDK    .S1     16384,A0          ; @@|80|

           SHR     .S1     A0,15,A3          ; @@|80|
           STH     .D1T1   A3,*++A4(10)      ; @@|80|
           STH     .D1T2   B4,*++A4(12)      ; @@|80|
;** --------------------------------------------------------------------------*

           MV      .L1X    B12,A0            ; |100|
||         ZERO    .S1     A6                ; |109|

           STW     .D1T1   A6,*++A0(24)      ; |109|
||         MV      .L1X    B13,A5            ; |109|

           LDH     .D1T1   *A5,A0            ; |101|
;** --------------------------------------------------------------------------*
L20:
           LDH     .D1T1   *A5,A3            ; |111|
;** --------------------------------------------------------------------------*
```

```
L21:
          NOP             1
;** --------------------------------------------------------------------------*
L22:

          LDH     .D1T1   *+A5(2),A8        ; |112|
||        MV      .L1X    B12,A4            ; |112|

          STW     .D1T1   A6,*+A4(20)       ; |109|
          STW     .D1T1   A6,*A4            ; |109|

          MPY     .M1     A3,A0,A0          ; |111|
||        STW     .D1T1   A6,*+A4(28)       ; |109|

          STW     .D1T1   A6,*+A4(4)        ; |109|
||        MV      .L1X    B12,A7            ; |113|

          STW     .D1T1   A0,*A7            ; |111|
||        MPY     .M1     A8,A8,A4          ; |113|

          STW     .D1T1   A6,*+A7(8)        ; |109|

          LDH     .D1T1   *A5,A3            ; |113|
||        MV      .L1X    B12,A4            ; |109|
||        ADD     .S1     A0,A4,A0          ; |113|

          MV      .L1X    B12,A0            ; |115|
||        STW     .D1T1   A0,*A4            ; |113|

          LDW     .D1T1   *+A0(4),A0        ; |113|
          LDH     .D1T1   *A5,A9            ; |115|
          STW     .D1T1   A6,*+A4(12)       ; |109|

          LDH     .D1T1   *+A5(4),A8        ; |114|
||        MPY     .M1     A3,A8,A3          ; |113|

          LDH     .D1T1   *+A5(2),A7        ; |115|

          LDW     .D1T1   *+A4(8),A1        ; |115|
||        MV      .L1X    B12,A3            ; |109|
||        ADD     .S1     A3,A0,A0          ; |113|

          MV      .L1X    B12,A0            ; |116|
||        STW     .D1T1   A0,*+A3(4)        ; |113|

          STW     .D1T1   A6,*+A0(16)       ; |109|
          LDW     .D1T1   *A0,A10           ; |115|

          MPY     .M1     A9,A8,A0          ; |115|
||        LDW     .D1T1   *+A0(4),A2        ; |115|
```

```
           STW     .D1T1   A6,*+A4(32)       ; |109|

           ADD     .L1     A0,A1,A0          ; |115|
||         MV      .S1X    B12,A6            ; |117|
||         MPY     .M1     A8,A8,A4          ; |115|

           STW     .D1T1   A0,*+A6(8)        ; |115|

           LDH     .D1T1   *+A5(6),A3        ; |116|
||         MV      .L1X    B12,A7            ; |117|
||         MPY     .M1     A7,A8,A0          ; |115|
||         ADD     .S1     A4,A10,A4         ; |115|

           STW     .D1T1   A4,*A7            ; |115|

           LDH     .D1T1   *+A5(2),A1        ; |117|
||         MV      .L1X    B12,A4            ; |115|
||         ADD     .S1     A0,A2,A0          ; |115|

           STW     .D1T1   A0,*+A4(4)        ; |115|
           LDW     .D1T1   *+A6(8),A6        ; |117|

           LDH     .D1T1   *+A5(4),A9        ; |117|
||         MV      .L1X    B12,A0            ; |115|

           LDW     .D1T1   *A0,A0            ; |117|
           MPY     .M1     A1,A3,A7          ; |117|
           LDW     .D1T1   *+A4(4),A4        ; |117|

           MPY     .M1     A3,A3,A7          ; |117|
||         ADD     .L1     A7,A6,A6          ; |117|

           MV      .L1X    B12,A8            ; |117|

           MPY     .M1     A9,A3,A6          ; |117|
||         ADD     .L1     A7,A0,A7          ; |117|
||         STW     .D1T1   A6,*+A8(8)        ; |117|

           STW     .D1T1   A7,*A8            ; |117|

           LDH     .D1T1   *+A5(8),A0        ; |118|
||         MV      .L1X    B12,A6            ; |117|
||         ADD     .S1     A6,A4,A4          ; |117|

           MV      .L1X    B12,A4            ; |117|
||         STW     .D1T1   A4,*+A6(4)        ; |117|

           LDW     .D1T1   *A4,A4            ; |119|
           NOP             2
           MPY     .M1     A0,A0,A6          ; |119|
           MV      .L1X    B12,A7
```

```
           ADD     .L1     A6,A4,A6          ; |119|
||         LDW     .D1T1   *+A7(4),A4        ; |119|

           STW     .D1T1   A6,*A7            ; |119|

           LDH     .D1T1   *+A5(4),A7        ; |119|
||         MV      .L1X    B12,A6            ; |117|

           LDW     .D1T1   *+A6(8),A6        ; |119|
           LDH     .D1T1   *A5,A9            ; |117|
           LDW     .D1T1   *+A8(12),A8       ; |117|
           NOP             1
           MPY     .M1     A7,A0,A7          ; |119|
           NOP             1

           ADD     .S1     A7,A6,A3          ; |119|
||         MPY     .M1     A9,A3,A6          ; |117|
||         MV      .L1X    B12,A7            ; |119|

           STW     .D1T1   A3,*+A7(8)        ; |119|

           ADD     .L1     A6,A8,A3          ; |117|
||         LDH     .D1T1   *+A5(2),A6        ; |119|

           STW     .D1T1   A3,*+A7(12)       ; |117|
||         MV      .L1X    B12,A3            ; |117|

           LDW     .D1T1   *+A3(12),A3       ; |119|
           NOP             2
           MPY     .M1     A6,A0,A6          ; |119|
           NOP             1

           ADD     .L1     A6,A3,A3          ; |119|
||         LDH     .D1T1   *A5,A6            ; |119|

           STW     .D1T1   A3,*+A7(12)       ; |119|
||         MV      .L1X    B12,A3            ; |119|

           LDW     .D1T1   *+A3(16),A3       ; |119|
           NOP             2
           MPY     .M1     A6,A0,A6          ; |119|
           LDH     .D1T1   *+A5(6),A7        ; |119|

           ADD     .S1     A6,A3,A3          ; |119|
||         MV      .L1X    B12,A6            ; |119|

           STW     .D1T1   A3,*+A6(16)       ; |119|
           LDH     .D1T1   *+A5(10),A3       ; |120|
           LDW     .D1T1   *A6,A6            ; |121|
           MPY     .M1     A7,A0,A0          ; |119|
```

```
           MV      .S1X    B12,A7           ; |121|
           NOP             1

           ADD     .L1     A0,A4,A4         ; |119|
||         MPY     .M1     A3,A3,A0         ; |121|

           STW     .D1T1   A4,*+A7(4)       ; |119|

           LDH     .D1T1   *+A5(8),A4       ; |121|
||         ADD     .S1     A0,A6,A0         ; |121|
||         MV      .L1X    B12,A6           ; |121|

           STW     .D1T1   A0,*A6           ; |121|
||         MV      .L1X    B12,A0           ; |121|

           LDW     .D1T1   *+A0(4),A0       ; |121|
           NOP             2
           MPY     .M1     A4,A3,A4         ; |121|
           NOP             1

           ADD     .S1     A4,A0,A0         ; |121|
||         MV      .L1X    B12,A4           ; |121|

           STW     .D1T1   A0,*+A4(4)       ; |121|

           MV      .L1X    B12,A0           ; |121|
||         LDH     .D1T1   *+A5(6),A9       ; |121|

           LDW     .D1T1   *+A0(8),A8       ; |121|
           LDH     .D1T1   *+A5(4),A0       ; |121|
           LDW     .D1T1   *+A4(12),A4      ; |121|
           NOP             3
           MPY     .M1     A0,A3,A0         ; |121|
           NOP             1

           ADD     .S1     A0,A4,A0         ; |121|
||         MV      .L1X    B12,A4           ; |121|

           STW     .D1T1   A0,*+A4(12)      ; |121|

           LDH     .D1T1   *+A5(2),A1       ; |121|
||         MV      .L1X    B12,A0           ; |121|

           LDW     .D1T1   *+A0(16),A4      ; |121|
           LDH     .D1T1   *A5,A0           ; |121|
           LDW     .D1T1   *+A6(20),A6      ; |121|
           NOP             3
           MPY     .M1     A0,A3,A0         ; |121|
           NOP             1

           ADD     .S1     A0,A6,A0         ; |121|
```

```
||         MV      .L1X    B12,A6           ; |121|

           STW     .D1T1   A0,*+A6(20)      ; |121|
           LDH     .D1T1   *+A5(12),A0      ; |122|
           LDW     .D1T1   *A6,A6           ; |123|
           LDH     .D1T1   *+A5(10),A2      ; |123|
           LDW     .D1T1   *+A7(4),A7       ; |123|
           MPY     .M1     A9,A3,A9         ; |121|
           NOP             2

           ADD     .S1     A9,A8,A9         ; |121|
||         MPY     .M1     A2,A0,A8         ; |123|
||         MV      .L1X    B12,A2           ; |123|

           STW     .D1T1   A9,*+A2(8)       ; |121|
||         MV      .L1X    B12,A9           ; |121|

           ADD     .L1     A8,A7,A8         ; |123|
||         LDW     .D1T1   *+A9(8),A7       ; |123|

           STW     .D1T1   A8,*+A9(4)       ; |123|
           LDH     .D1T1   *+A5(6),A8       ; |123|
           LDW     .D1T1   *+A9(12),A9      ; |123|
           MPY     .M1     A1,A3,A3         ; |121|
           NOP             2
           MPY     .M1     A8,A0,A8         ; |123|

           ADD     .S1     A3,A4,A3         ; |121|
||         MV      .L1X    B12,A4           ; |121|

           STW     .D1T1   A3,*+A4(16)      ; |121|
||         ADD     .L1     A8,A9,A3         ; |123|

           STW     .D1T1   A3,*+A4(12)      ; |123|
||         MV      .L1X    B12,A3           ; |123|

           LDW     .D1T1   *+A3(16),A8      ; |123|
           LDH     .D1T1   *+A5(2),A4       ; |123|
           LDW     .D1T1   *+A3(20),A3      ; |123|
           MV      .S1X    B12,A1           ; |123|
           NOP             2
           MPY     .M1     A4,A0,A4         ; |123|
           MV      .L1X    B12,A9

           ADD     .L1     A4,A3,A3         ; |123|
||         LDW     .D1T1   *+A9(24),A9      ; |123|
||         MPY     .M1     A0,A0,A4         ; |123|

           STW     .D1T1   A3,*+A1(20)      ; |123|

           LDH     .D1T1   *+A5(14),A1      ; |124|
```

```
||         ADD     .S1     A4,A6,A3          ; |123|
||         MV      .L1X    B12,A4            ; |123|

           STW     .D1T1   A3,*A4            ; |123|
||         MV      .L1X    B12,A3            ; |123|

           LDW     .D1T1   *A3,A6            ; |125|
           LDH     .D1T1   *+A5(8),A3        ; |123|
           NOP             1
           MPY     .M1     A1,A1,A4          ; |125|
           NOP             1

           ADD     .S1     A4,A6,A4          ; |125|
||         MV      .L1X    B12,A6            ; |125|

           STW     .D1T1   A4,*A6            ; |125|
||         MPY     .M1     A3,A0,A4          ; |123|
||         MV      .L1X    B12,A3            ; |123|

           LDW     .D1T1   *+A3(4),A3        ; |125|

           ADD     .S1     A4,A7,A6          ; |123|
||         LDH     .D1T1   *+A5(10),A4       ; |125|
||         MV      .L1X    B12,A7            ; |125|

           STW     .D1T1   A6,*+A7(8)        ; |123|
||         MV      .L1X    B12,A6            ; |123|

           LDW     .D1T1   *+A6(8),A6        ; |125|
           LDH     .D1T1   *A5,A7            ; |123|
           NOP             1
           MPY     .M1     A4,A1,A4          ; |125|
           NOP             1

           ADD     .L1     A4,A6,A6          ; |125|
||         LDH     .D1T1   *+A5(4),A4        ; |123|

           STW     .D1T1   A6,*+A2(8)        ; |125|
           LDH     .D1T1   *+A5(8),A11       ; |125|

           MPY     .M1     A7,A0,A7          ; |123|
||         LDW     .D1T1   *+A2(12),A10      ; |125|

           LDH     .D1T1   *+A5(2),A2        ; |125|

           LDH     .D1T1   *+A5(6),A6        ; |125|
||         MPY     .M1     A4,A0,A4          ; |123|
||         ADD     .S1     A7,A9,A0          ; |123|
||         MV      .L1X    B12,A9            ; |125|

           STW     .D1T1   A0,*+A9(24)       ; |123|
```

```
           LDH     .D1T1   *+A5(4),A7       ; |125|
||         MPY     .M1     A11,A1,A0       ; |125|

           LDW     .D1T1   *+A9(24),A9     ; |125|
||         ADD     .S1     A4,A8,A4        ; |123|
||         MV      .L1X    B12,A8          ; |125|

           ADD     .L1     A0,A10,A0       ; |125|
||         STW     .D1T1   A4,*+A8(16)     ; |123|
||         MV      .S1X    B12,A4          ; |123|

           STW     .D1T1   A0,*+A4(12)     ; |125|
||         MV      .L1X    B12,A0          ; |125|

           LDW     .D1T1   *+A0(16),A8     ; |125|
||         MPY     .M1     A2,A1,A0        ; |125|

           MVK     .S2     0x2,B1          ; init prolog collapse predicate
           ADD     .L1     A0,A9,A0        ; |125|
           STW     .D1T1   A0,*+A4(24)     ; |125|
           LDW     .D1T2   *+A4(24),B4
           LDH     .D1T1   *+A5(12),A2     ; |125|
           LDW     .D1T1   *+A4(20),A10    ; |125|

           LDH     .D1T1   *A5++(14),A9    ; |125|
||         MV      .L1X    B12,A0          ; |125|
||         MPY     .M1     A6,A1,A4        ; |125|

           LDW     .D1T1   *+A0(28),A7     ; |125|
||         MPY     .M1     A7,A1,A0        ; |125|
||         MV      .L1X    B12,A6

           LDW     .D1T2   *+A6(8),B7

           ADD     .S1     A4,A8,A4        ; |125|
||         MPY     .M1     A2,A1,A6        ; |125|
||         MV      .L1X    B12,A8          ; |125|

           ADD     .L1     A0,A10,A0       ; |125|
||         STW     .D1T1   A4,*+A8(16)     ; |125|
||         MV      .S1X    B12,A2          ; |125|

           STW     .D1T1   A0,*+A2(20)     ; |125|
||         ADD     .L1     A6,A3,A3        ; |125|
||         MV      .S1X    B12,A4          ; |125|
||         MPY     .M1     A9,A1,A0        ; |125|

           STW     .D1T1   A3,*+A4(4)      ; |125|
||         MV      .L1X    B12,A3          ; |125|
```

```
           ADD     .L1     A0,A7,A0          ; |125|
||         LDW     .D1T1   *+A3(32),A2

           MV      .L2X    A0,B10
||         STW     .D1T1   A0,*+A3(28)       ; |125|
||         MV      .L1X    B12,A0

           LDW     .D1T2   *+A0(20),B6
           LDW     .D1T1   *+A3(4),A7
           LDW     .D1T1   *A3,A6
           LDW     .D1T1   *+A3(16),A4
           LDW     .D1T1   *+A3(12),A9       ;
           MVK     .S1     0x1,A1            ; init prolog collapse predicate

           MV      .L2X    A5,B3
||         MV      .L1X    B7,A8
||         MVK     .S2     151,B0            ;

;** --------------------------------------------------------------------------*
L23:    ; PIPED LOOP PROLOG
;** --------------------------------------------------------------------------*
L24:    ; PIPED LOOP KERNEL

           MV      .L2     B2,B11            ; Inserted to split a long life
|| [!B1]   ADD     .L1     A13,A6,A6         ; |131|
|| [ B0]   B       .S2     L24               ; |134|
||         MPY     .M2X    B9,A0,B8          ; |133|
||         MPY     .M1     A11,A0,A13        ; |132|
|| [!A1]   LDH     .D1T2   *-A5(8),B2        ; @|132|
|| [!A1]   LDH     .D2T1   *B3++,A14         ; @|132|

   [!B1]   ADD     .L1     A15,A7,A7         ; |132|
||         MPY     .M1     A12,A0,A15        ; |133|
|| [!A1]   LDH     .D2T1   *-B3(6),A11       ; @|132|
|| [!A1]   LDH     .D1T2   *-A5(10),B9       ; @|133|

   [!B1]   ADD     .L2     B8,B6,B6          ; |133|
|| [!B1]   ADD     .L1     A13,A9,A9         ; |132|
||         MPY     .M2X    B5,A0,B8          ; |133|
||         MPY     .M1     A10,A0,A13        ; |132|
|| [!A1]   LDH     .D1T1   *-A5(16),A12      ; @|133|

           MPY     .M1X    B11,A0,A0         ; |132|
||         MPY     .M2X    B7,A0,B11         ; |133|
|| [!B1]   ADD     .L1     A15,A2,A2         ; |133|
|| [!A1]   LDH     .D1T2   *-A5(14),B5       ; @|133|
|| [!A1]   LDH     .D2T1   *-B3(4),A10       ; @|132|

   [ A1]   SUB     .S1     A1,1,A1           ;
|| [!B1]   ADD     .L2     B8,B10,B10        ; |133|
|| [!B1]   ADD     .L1     A13,A8,A8         ; |132|
```

```
||         MPY     .M1     A3,A3,A13         ; @|131|
|| [!A1]   LDH     .D1T2   *-A5(12),B7       ; @|133|

   [ B1]   SUB     .S2     B1,1,B1           ;
|| [!B1]   ADD     .S1     A0,A4,A4          ; |132|
|| [!B1]   ADD     .D2     B11,B4,B4         ; |133|
||         MV      .L1     A3,A0             ; @Inserted to split a long life
|| [ B0]   SUB     .L2     B0,1,B0           ; @|134|
||         MPY     .M1     A14,A3,A15        ; @|132|
||         LDH     .D1T1   *++A5,A3          ; @@|131|

;** --------------------------------------------------------------------------*
L25:    ; PIPED LOOP EPILOG

           MV      .L2     B2,B11            ; @Inserted to split a long life
||         ADD     .L1     A13,A6,A6         ; @|131|
||         MPY     .M2X    B9,A0,B8          ; @|133|
||         MPY     .M1     A11,A0,A13        ; @|132|
||         LDH     .D1T2   *-A5(8),B2        ; @@|132|
||         LDH     .D2T1   *B3++,A14         ; @@|132|

           ADD     .L1     A15,A7,A7         ; @|132|
||         MPY     .M1     A12,A0,A15        ; @|133|
||         LDH     .D2T1   *-B3(6),A11       ; @@|132|
||         LDH     .D1T2   *-A5(10),B9       ; @@|133|

           ADD     .L2     B8,B6,B6          ; @|133|
||         ADD     .L1     A13,A9,A9         ; @|132|
||         MPY     .M2X    B5,A0,B8          ; @|133|
||         MPY     .M1     A10,A0,A13        ; @|132|
||         LDH     .D1T1   *-A5(16),A12      ; @@|133|

           MPY     .M1X    B11,A0,A0         ; @|132|
||         MPY     .M2X    B7,A0,B11         ; @|133|
||         ADD     .L1     A15,A2,A2         ; @|133|
||         LDH     .D1T2   *-A5(14),B5       ; @@|133|
||         LDH     .D2T1   *-B3(4),A10       ; @@|132|

           ADD     .L2     B8,B10,B10        ; @|133|
||         ADD     .L1     A13,A8,A8         ; @|132|
||         MPY     .M1     A3,A3,A13         ; @@|131|
||         LDH     .D1T2   *-A5(12),B7       ; @@|133|

           ADD     .S1     A0,A4,A4          ; @|132|
||         ADD     .D2     B11,B4,B4         ; @|133|
||         MV      .L1     A3,A0             ; @@Inserted to split a long life
||         MPY     .M1     A14,A3,A15        ; @@|132|

           MV      .L2     B2,B11            ; @@Inserted to split a long life
||         ADD     .L1     A13,A6,A6         ; @@|131|
||         MPY     .M2X    B9,A0,B8          ; @@|133|
```

```
||         MPY     .M1     A11,A0,A13        ; @@|132|

           ADD     .L1     A15,A7,A7         ; @@|132|
||         MPY     .M1     A12,A0,A15        ; @@|133|

           ADD     .L2     B8,B6,B6          ; @@|133|
||         ADD     .L1     A13,A9,A9         ; @@|132|
||         MPY     .M2X    B5,A0,B8          ; @@|133|
||         MPY     .M1     A10,A0,A13        ; @@|132|

           MPY     .M1X    B11,A0,A0         ; @@|132|
||         MPY     .M2X    B7,A0,B11         ; @@|133|
||         ADD     .L1     A15,A2,A2         ; @@|133|

           ADD     .L2     B8,B10,B10        ; @@|133|
||         ADD     .L1     A13,A8,A8         ; @@|132|

           ADD     .S1     A0,A4,A4          ; @@|132|
||         ADD     .D2     B11,B4,B4         ; @@|133|

;** --------------------------------------------------------------------------*
           MV      .L1X    B12,A0
           STW     .D1T1   A7,*+A0(4)
           STW     .D1T2   B4,*+A0(24)
           STW     .D1T2   B10,*+A0(28)
           STW     .D1T1   A4,*+A0(16)
           STW     .D1T1   A2,*+A0(32)

           STW     .D1T2   B6,*+A0(20)
||         MV      .L1X    B12,A3            ; |136|

           STW     .D1T1   A6,*A3
           LDW     .D1T1   *+A0(24),A0       ; |136|

           LDW     .D1T1   *+A3(28),A5       ; |136|
||         MV      .L2X    A9,B5

           STW     .D1T2   B5,*+A3(12)
           LDW     .D1T1   *+A3(16),A7       ; |136|

           LDW     .D1T1   *+A3(32),A8       ; |136|
||         MV      .L2X    A8,B7

           LDW     .D1T1   *+A3(20),A9       ; |136|
||         SHL     .S1     A0,1,A0           ; |136|

           STW     .D1T1   A0,*+A3(24)       ; |136|
           LDW     .D1T1   *+A3(4),A6        ; |136|

           STW     .D1T2   B7,*+A3(8)
||         MV      .L1X    B12,A0            ; |136|
```

```
           LDW     .D1T1   *+A0(12),A1       ; |136|
           LDW     .D1T1   *+A0(8),A2        ; |136|

           LDW     .D1T1   *A0,A8            ; |136|
||         SHL     .S1     A8,1,A0           ; |136|

           SHL     .S1     A9,1,A0           ; |136|
||         STW     .D1T1   A0,*+A3(32)       ; |136|

           SHL     .S1     A5,1,A0           ; |136|
||         STW     .D1T1   A0,*+A3(20)       ; |136|

           SHL     .S1     A7,1,A0           ; |136|
||         STW     .D1T1   A0,*+A3(28)       ; |136|

           SHL     .S1     A2,1,A0           ; |136|
||         STW     .D1T1   A0,*+A3(16)       ; |136|

           SHL     .S1     A6,1,A0           ; |136|
||         STW     .D1T1   A0,*+A3(8)        ; |136|

           SHL     .S1     A1,1,A0           ; |136|
||         STW     .D1T1   A0,*+A3(4)        ; |136|

           STW     .D1T1   A0,*+A3(12)       ; |136|
           LDW     .D2T1   *+SP(4),A3        ; |136|
           NOP             4
           CMPGT   .L2X    A3,0,B0           ; |141|
   [!B0]   B       .S1     L30               ; |141|
           SHL     .S1     A8,1,A0           ; |136|
           MV      .L1X    B12,A5            ; |136|
           CMPGT   .L1     A3,4,A1           ; |142|
           STW     .D1T1   A0,*A5            ; |136|
           NOP             1
           ; BRANCH OCCURS                   ; |141|
;** --------------------------------------------------------------------------*
   [!A1]   B       .S1     L26               ; |142|
           MVKL    .S2     RL4,B3            ; |142|

           MVKH    .S2     RL4,B3            ; |142|
|| [!A1]   SUB     .L1X    B13,4,A0          ;

   [!A1]   LDH     .D1T1   *++A0(4),A3       ; |143|

   [!A1]   LDW     .D2T1   *+SP(4),A0
|| [!A1]   MV      .L2X    A0,B13            ; |143|

           NOP             1
           ; BRANCH OCCURS                   ; |142|
;** --------------------------------------------------------------------------*
```

```
           B       .S1     __abort_msg       ; |142|
           MVKL    .S1     SL2+0,A4          ; |142|
           MVKH    .S1     SL2+0,A4          ; |142|
           NOP             3
RL4:       ; CALL OCCURS                     ; |142|
           SUB     .L1X    B13,4,A0          ;
           LDH     .D1T1   *++A0(4),A3       ; |143|

           LDW     .D2T1   *+SP(4),A0
||         MV      .L2X    A0,B13            ; |143|

           NOP             1
;** --------------------------------------------------------------------------*
L26:
           NOP             3

           SHL     .S1     A3,A0,A3          ; |143|
||         MV      .L1X    B13,A0            ; |143|

           STH     .D1T1   A3,*A0            ; |143|
           LDH     .D1T1   *+A0(2),A3        ; |143|
           LDW     .D2T1   *+SP(4),A0
           LDW     .D2T1   *+SP(4),A7
           NOP             3

           SHL     .S1     A3,A0,A3          ; |143|
||         MV      .L1X    B13,A0            ; |143|

           STH     .D1T1   A3,*+A0(2)        ; |143|
           LDH     .D1T1   *++A0(4),A3       ; |143|
           ADD     .L2X    4,A0,B9           ;
           MV      .L2X    A0,B13            ;
           MV      .S1     A0,A5             ;
           MV      .L1     A7,A0

           MV      .L1X    B13,A0            ; |143|
||         SHL     .S1     A3,A0,A3          ; |143|

           STH     .D1T1   A3,*A0            ; |143|
           LDH     .D1T1   *+A0(2),A3        ; |143|
           MV      .L1     A7,A0
           MVK     .S1     0x2,A1            ; init prolog collapse predicate
           MVK     .S2     78,B0             ;
           MV      .L2X    A7,B4

           SHL     .S1     A3,A0,A3          ; |143|
||         MV      .L1X    B13,A0            ; |143|

           STH     .D1T1   A3,*+A0(2)        ; |143|
||         SUBAH   .D2     B0,9,B0
||         MVK     .S1     0x1,A2            ; init prolog collapse predicate
```

```
;** --------------------------------------------------------------------------*
L27:    ; PIPED LOOP PROLOG
           LDH     .D2T2   *B9++(12),B5      ; |143|
           NOP             5

           SHL     .S2     B5,B4,B7          ; |143|
||         LDH     .D2T2   *B9++(12),B5      ; @|143|

           STH     .D2T2   B7,*-B9(24)       ; |143|
           LDH     .D2T2   *-B9(22),B8       ; |143|
           NOP             3

           SHL     .S2     B5,B4,B7          ; @|143|
||         LDH     .D2T2   *B9++(12),B5      ; @@|143|

           STH     .D2T2   B7,*-B9(24)       ; @|143|

           SHL     .S2     B8,B4,B7          ; |143|
||         LDH     .D2T2   *-B9(22),B8       ; @|143|

           STH     .D2T2   B7,*-B9(34)       ; |143|
           LDH     .D2T2   *-B9(32),B6       ; |143|
           NOP             1

           SHL     .S2     B5,B4,B7          ; @@|143|
||         LDH     .D2T2   *B9++(12),B5      ; @@@|143|

           STH     .D2T2   B7,*-B9(24)       ; @@|143|

           SHL     .S2     B8,B4,B7          ; @|143|
||         LDH     .D2T2   *-B9(22),B8       ; @@|143|

           STH     .D2T2   B7,*-B9(34)       ; @|143|

           SHL     .S2     B6,B4,B7          ; |143|
||         LDH     .D2T2   *-B9(32),B6       ; @|143|

           STH     .D2T2   B7,*-B9(44)       ; |143|

           LDH     .D1T1   *+A5(10),A4       ; |143|
||         SHL     .S2     B5,B4,B7          ; @@@|143|
||         LDH     .D2T2   *B9++(12),B5      ; @@@@|143|

           STH     .D2T2   B7,*-B9(24)       ; @@@|143|

           SHL     .S2     B8,B4,B7          ; @@|143|
||         LDH     .D2T2   *-B9(22),B8       ; @@@|143|

           STH     .D2T2   B7,*-B9(34)       ; @@|143|
```

```
           SHL     .S2     B6,B4,B7          ; @|143|
||         LDH     .D2T2   *-B9(32),B6       ; @@|143|

           SHL     .S1     A4,A7,A6          ; |143|
||         STH     .D2T2   B7,*-B9(44)       ; @|143|

;** --------------------------------------------------------------------------*
L28:    ; PIPED LOOP KERNEL

   [ B0]   B       .S1     L28               ; |143|
||         LDH     .D1T1   *+A5(22),A4       ; @@@|143|
||         SHL     .S2     B5,B4,B7          ; @@@@@@|143|
||         LDH     .D2T2   *B9++(12),B5      ; @@@@@@@|143|

           STH     .D1T1   A6,*+A5(10)       ; @@|143|
||         STH     .D2T2   B7,*-B9(24)       ; @@@@@@|143|

           SHL     .S1     A3,A7,A6          ; @|143|
||         LDH     .D1T1   *++A5(12),A3      ; @@|143|
||         SHL     .S2     B8,B4,B7          ; @@@@@|143|
||         LDH     .D2T2   *-B9(22),B8       ; @@@@@@|143|

   [!A2]   STH     .D1T1   A6,*-A5(12)       ; @|143|
||         STH     .D2T2   B7,*-B9(34)       ; @@@@@|143|

   [ A2]   SUB     .L1     A2,1,A2           ;
||         SHL     .S1     A0,A7,A6          ; |143|
|| [!A2]   LDH     .D1T1   *-A5(10),A0       ; @|143|
||         SHL     .S2     B6,B4,B7          ; @@@@|143|
||         LDH     .D2T2   *-B9(32),B6       ; @@@@@|143|

   [ A1]   SUB     .L1     A1,1,A1           ;
|| [!A1]   STH     .D1T1   A6,*-A5(22)       ; |143|
|| [ B0]   SUB     .L2     B0,3,B0           ; @|143|
||         SHL     .S1     A4,A7,A6          ; @@@|143|
||         STH     .D2T2   B7,*-B9(44)       ; @@@@|143|

;** --------------------------------------------------------------------------*
L29:    ; PIPED LOOP EPILOG

           LDH     .D1T1   *+A5(22),A4       ; @@@@|143|
||         SHL     .S2     B5,B4,B7          ; @@@@@@@|143|

           STH     .D1T1   A6,*+A5(10)       ; @@@|143|
||         STH     .D2T2   B7,*-B9(12)       ; @@@@@@@|143|

           SHL     .S1     A3,A7,A6          ; @@|143|
||         LDH     .D1T1   *++A5(12),A3      ; @@@|143|
||         SHL     .S2     B8,B4,B7          ; @@@@@@|143|
||         LDH     .D2T2   *-B9(10),B8       ; @@@@@@@|143|
```

```
           STH     .D1T1   A6,*-A5(12)       ; @@|143|
||         STH     .D2T2   B7,*-B9(22)       ; @@@@@@|143|

           SHL     .S1     A0,A7,A6          ; @|143|
||         LDH     .D1T1   *-A5(10),A0       ; @@|143|
||         SHL     .S2     B6,B4,B7          ; @@@@@|143|
||         LDH     .D2T2   *-B9(20),B6       ; @@@@@@|143|

           STH     .D1T1   A6,*-A5(22)       ; @|143|
||         SHL     .S1     A4,A7,A6          ; @@@@|143|
||         STH     .D2T2   B7,*-B9(32)       ; @@@@@|143|

           LDH     .D1T1   *+A5(22),A4       ; @@@@@|143|
           STH     .D1T1   A6,*+A5(10)       ; @@@@|143|

           SHL     .S1     A3,A7,A6          ; @@@|143|
||         LDH     .D1T1   *++A5(12),A3      ; @@@@|143|
||         SHL     .S2     B8,B4,B7          ; @@@@@@@|143|

           STH     .D1T1   A6,*-A5(12)       ; @@@|143|
||         STH     .D2T2   B7,*-B9(10)       ; @@@@@@@|143|

           SHL     .S1     A0,A7,A6          ; @@|143|
||         LDH     .D1T1   *-A5(10),A0       ; @@@|143|
||         SHL     .S2     B6,B4,B7          ; @@@@@@|143|
||         LDH     .D2T2   *-B9(8),B6        ; @@@@@@@|143|

           STH     .D1T1   A6,*-A5(22)       ; @@|143|
||         SHL     .S1     A4,A7,A6          ; @@@@@|143|
||         STH     .D2T2   B7,*-B9(20)       ; @@@@@@|143|

           LDH     .D1T1   *+A5(22),A4       ; @@@@@@|143|
           STH     .D1T1   A6,*+A5(10)       ; @@@@@|143|

           SHL     .S1     A3,A7,A6          ; @@@@|143|
||         LDH     .D1T1   *++A5(12),A3      ; @@@@@|143|

           STH     .D1T1   A6,*-A5(12)       ; @@@@|143|

           SHL     .S1     A0,A7,A6          ; @@@|143|
||         LDH     .D1T1   *-A5(10),A0       ; @@@@|143|
||         SHL     .S2     B6,B4,B7          ; @@@@@@@|143|

           STH     .D1T1   A6,*-A5(22)       ; @@@|143|
||         SHL     .S1     A4,A7,A6          ; @@@@@@|143|
||         STH     .D2T2   B7,*-B9(8)        ; @@@@@@@|143|

           LDH     .D1T1   *+A5(22),A4       ; @@@@@@@|143|
           STH     .D1T1   A6,*+A5(10)       ; @@@@@@|143|

           SHL     .S1     A3,A7,A6          ; @@@@@|143|
```

```
||         LDH     .D1T1   *++A5(12),A3      ; @@@@@@|143|

           STH     .D1T1   A6,*-A5(12)       ; @@@@@|143|

           SHL     .S1     A0,A7,A6          ; @@@@|143|
||         LDH     .D1T1   *-A5(10),A0       ; @@@@@|143|

           STH     .D1T1   A6,*-A5(22)       ; @@@@|143|
||         SHL     .S1     A4,A7,A6          ; @@@@@@@|143|

           NOP             1
           STH     .D1T1   A6,*+A5(10)       ; @@@@@@@|143|

           SHL     .S1     A3,A7,A6          ; @@@@@@|143|
||         LDH     .D1T1   *++A5(12),A3      ; @@@@@@@|143|

           STH     .D1T1   A6,*-A5(12)       ; @@@@@@|143|

           SHL     .S1     A0,A7,A6          ; @@@@@|143|
||         LDH     .D1T1   *-A5(10),A0       ; @@@@@@|143|

           STH     .D1T1   A6,*-A5(22)       ; @@@@@|143|
           NOP             2
           SHL     .S1     A3,A7,A6          ; @@@@@@@|143|
           STH     .D1T1   A6,*A5            ; @@@@@@@|143|

           SHL     .S1     A0,A7,A6          ; @@@@@@|143|
||         LDH     .D1T1   *+A5(2),A0        ; @@@@@@@|143|

           STH     .D1T1   A6,*-A5(10)       ; @@@@@@|143|
           NOP             4
           SHL     .S1     A0,A7,A6          ; @@@@@@@|143|
           STH     .D1T1   A6,*+A5(2)        ; @@@@@@@|143|
;** --------------------------------------------------------------------------*
L30:
           LDW     .D2T2   *+SP(32),B3       ; |145|
           LDW     .D2T2   *+SP(40),B11      ; |145|
           LDW     .D2T2   *+SP(36),B10      ; |145|
           LDW     .D2T1   *+SP(28),A15      ; |145|
           LDW     .D2T1   *+SP(24),A14      ; |145|
           LDW     .D2T1   *+SP(20),A13      ; |145|
           LDW     .D2T1   *+SP(16),A12      ; |145|
           LDW     .D2T1   *+SP(12),A11      ; |145|
           LDW     .D2T1   *+SP(8),A10       ; |145|

           B       .S2     B3                ; |145|
||         LDW     .D2T2   *+SP(44),B12      ; |145|

           LDW     .D2T2   *++SP(48),B13     ; |145|
           NOP             4
           ; BRANCH OCCURS                   ; |145|
```

```
.sect ".text"
.global _gsm_encode

;******************************************************************************
;* FUNCTION NAME: _gsm_encode                                                 *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 216 Auto + 44 Save = 260 byte               *
;******************************************************************************
_gsm_encode:
;** --------------------------------------------------------------------------*
           ADDK    .S2     -264,SP           ; |13|
           ADDAW   .D2     SP,9,B7           ; |17|
           STW     .D2T2   B3,*+SP(248)      ; |13|
           STW     .D2T1   A11,*+SP(228)     ; |13|
           STW     .D2T1   A10,*+SP(224)     ; |13|
           STW     .D2T2   B10,*+SP(252)     ; |13|
           ADDAW   .D2     SP,11,B5          ; |17|
           STW     .D2T1   A15,*+SP(244)     ; |13|
           STW     .D2T1   A14,*+SP(240)     ; |13|
           STW     .D2T1   A13,*+SP(236)     ; |13|
           STW     .D2T1   A12,*+SP(232)     ; |13|

           B       .S1     _Gsm_Coder        ; |17|
||         STW     .D2T2   B11,*+SP(256)     ; |13|

           ADDAW   .D2     SP,7,B8           ; |17|
           ADDAW   .D2     SP,5,B6           ; |17|

           MV      .L1X    B7,A8             ; |17|
||         STW     .D2T2   B12,*+SP(260)     ; |13|

           ADD     .L1X    4,SP,A6           ; |17|
||         MVKL    .S2     RL0,B3            ; |17|
||         MV      .S1     A6,A11            ;
||         STW     .D2T2   B13,*+SP(264)     ; |13|

           MV      .L1X    B5,A10            ; |17|
||         ADDAW   .D2     SP,13,B10         ; |17|
||         MVKH    .S2     RL0,B3            ; |17|

RL0:       ; CALL OCCURS                     ; |17|
           LDH     .D2T2   *+SP(4),B8        ; |106|
           LDH     .D2T2   *+SP(6),B7        ; |108|
           LDH     .D2T2   *+SP(4),B6        ; |108|
```

```
           LDH     .D2T2   *+SP(10),B5       ; |110|
           MVK     .S2     -48,B9            ; |106|

           LDH     .D2T2   *+SP(8),B4        ; |110|
||         EXTU    .S2     B8,26,28,B8       ; |106|

           EXTU    .S2     B7,26,26,B7       ; |108|

           OR      .L2     B9,B8,B8          ; |106|
||         SHL     .S2     B6,6,B6           ; |108|

           OR      .L2     B7,B6,B7          ; |108|
||         STB     .D1T2   B8,*A11++         ; |106|

           STB     .D1T2   B7,*A11++         ; |108|
||         EXTU    .S2     B5,27,29,B6       ; |110|

           LDH     .D2T2   *+SP(12),B4       ; |112|
||         SHL     .S2     B4,3,B8           ; |110|

           LDH     .D2T2   *+SP(10),B5       ; |112|
||         OR      .L2     B6,B8,B6          ; |110|

           STB     .D1T2   B6,*A11++         ; |110|
           LDH     .D2T1   *+SP(14),A3       ; |112|
           NOP             2
           AND     .L2     15,B4,B4          ; |112|

           SHL     .S1X    B5,6,A0           ; |112|
||         SHL     .S2     B4,2,B4           ; |112|

           EXTU    .S1     A3,28,30,A3       ; |112|
||         OR      .L1X    B4,A0,A0          ; |112|

           OR      .L1     A3,A0,A0          ; |112|

           STB     .D1T1   A0,*A11++         ; |112|
||         LDH     .D2T2   *+SP(14),B4       ; |115|

           LDH     .D2T1   *+SP(16),A0       ; |115|
           LDH     .D2T2   *+SP(18),B5       ; |115|
           NOP             3

           EXTU    .S1     A0,29,26,A0       ; |115|
||         SHL     .S2     B4,6,B4           ; |115|

           OR      .L1X    A0,B4,A0          ; |115|
||         AND     .L2     7,B5,B4           ; |115|

           OR      .L1X    B4,A0,A0          ; |115|
           STB     .D1T1   A0,*A11++         ; |115|
```

```
           LDH     .D2T1   *+SP(20),A0       ; |118|
           LDH     .D2T2   *+SP(36),B4       ; |118|
           LDH     .D2T2   *+SP(28),B5       ; |120|
           NOP             3

           EXTU    .S2     B4,30,31,B4       ; |118|
||         SHL     .S1     A0,1,A0           ; |118|

           OR      .L1X    B4,A0,A0          ; |118|
           STB     .D1T1   A0,*A11++         ; |118|
           LDH     .D2T2   *+SP(36),B4       ; |120|
           LDH     .D2T2   *+SP(44),B6       ; |120|
           AND     .L2     3,B5,B5           ; |120|
           LDH     .D2T2   *+SP(54),B9       ; |123|
           NOP             1

           SHL     .S1X    B4,7,A0           ; |120|
||         SHL     .S2     B5,5,B4           ; |120|

           OR      .L1X    B4,A0,A0          ; |120|
||         EXTU    .S2     B6,26,27,B4       ; |120|

           OR      .L1X    B4,A0,A0          ; |120|

           STB     .D1T1   A0,*A11++         ; |120|
||         LDH     .D2T2   *+SP(52),B7       ; |123|

           LDH     .D2T2   *+SP(44),B8       ; |123|
           LDH     .D2T2   *+SP(56),B5       ; |123|
           LDH     .D2T2   *+SP(74),B6       ; |137|
           LDH     .D2T2   *+SP(62),B0       ; |130|
           AND     .L2     7,B7,B7           ; |123|

           SHL     .S1X    B8,7,A3           ; |123|
||         AND     .L2     7,B9,B8           ; |123|
||         SHL     .S2     B7,4,B7           ; |123|

           OR      .L1X    B7,A3,A3          ; |123|
||         SHL     .S2     B8,1,B7           ; |123|

           EXTU    .S2     B5,29,31,B5       ; |123|
||         OR      .L1X    B7,A3,A3          ; |123|
||         LDH     .D2T2   *+SP(66),B2       ; |133|

           OR      .L1X    B5,A3,A3          ; |123|
||         LDH     .D2T2   *+SP(64),B3       ; |130|

           LDH     .D2T2   *+SP(58),B4       ; |127|
||         STB     .D1T1   A3,*A11++         ; |123|

           LDH     .D2T2   *+SP(56),B6       ; |127|
```

```
||          EXTU    .S2     B6,29,26,B9       ; |137|

            LDH     .D2T2   *+SP(60),B8       ; |127|
            LDH     .D2T1   *+SP(72),A0       ; |137|
            LDH     .D2T2   *+SP(68),B5       ; |133|
            LDH     .D2T2   *+SP(70),B7       ; |133|
            SHL     .S1X    B0,5,A4           ; |130|

            SHL     .S2     B6,6,B8           ; |127|
||          AND     .L2     7,B8,B6           ; |127|

            SHL     .S1X    B2,7,A3           ; |133|
||          EXTU    .S2     B4,29,26,B4       ; |127|
||          AND     .L2     7,B3,B0           ; |130|
||          LDH     .D2T2   *+SP(66),B1       ; |130|

            LDH     .D2T2   *+SP(72),B2       ; |133|
||          SHL     .S1     A0,6,A0           ; |137|
||          SHL     .S2     B0,2,B5           ; |130|
||          AND     .L2     7,B5,B0           ; |133|

            SHL     .S2     B0,4,B0           ; |133|
||          OR      .L2     B4,B8,B8          ; |127|
||          LDH     .D2T2   *+SP(76),B9       ; |137|
||          OR      .L1X    B9,A0,A0          ; |137|

            OR      .L1X    B0,A3,A3          ; |133|
||          AND     .L2     7,B7,B4           ; |133|

            OR      .L1X    B5,A4,A4          ; |130|
||          SHL     .S2     B4,1,B4           ; |133|

            OR      .L1X    B4,A3,A3          ; |133|
||          OR      .L2     B6,B8,B6          ; |127|
||          EXTU    .S2     B1,29,30,B1       ; |130|

            STB     .D1T2   B6,*A11++         ; |127|
||          OR      .L1X    B1,A4,A4          ; |130|
||          EXTU    .S2     B2,29,31,B4       ; |133|

            STB     .D1T1   A4,*A11++         ; |130|
||          AND     .L2     7,B9,B4           ; |137|
||          OR      .L1X    B4,A3,A3          ; |133|

            OR      .L1X    B4,A0,A0          ; |137|
||          STB     .D1T1   A3,*A11++         ; |133|

            MVK     .S1     110,A0            ; |171|
||          STB     .D1T1   A0,*A11++         ; |137|

            ADD     .L1X    A0,SP,A0          ; |171|
```

```
           LDH     .D1T1   *A0,A9           ; |171|
           LDH     .D1T1   *+A0(6),A2      ; |174|
           LDH     .D1T1   *+A0(8),A0      ; |174|
           MVK     .S1     114,A5          ; |174|
           ADD     .L1X    A5,SP,A7        ; |174|
           NOP             2
           STW     .D2T1   A0,*+SP(156)    ; |174|
           LDH     .D1T1   *+A7(12),A0     ; |181|
           NOP             4
           STW     .D2T1   A0,*+SP(160)    ; |181|
           LDH     .D1T1   *+A7(6),A0      ; |177|
           NOP             4
           STW     .D2T1   A0,*+SP(164)    ; |177|
           LDH     .D1T1   *+A7(10),A0     ; |181|
           LDH     .D2T2   *+SP(38),B7     ; |140|
           NOP             3
           STW     .D2T1   A0,*+SP(168)    ; |181|
           LDH     .D2T2   *+SP(22),B5     ; |140|
           LDH     .D1T1   *+A7(10),A0     ; |177|
           LDH     .D2T2   *+SP(30),B8     ; |142|
           LDH     .D2T2   *+SP(38),B9     ; |142|

           LDH     .D2T2   *+SP(46),B6     ; |142|
||         EXTU    .S2     B7,30,31,B7     ; |140|

           SHL     .S2     B5,1,B5         ; |140|

           STW     .D2T1   A0,*+SP(172)    ; |177|
||         OR      .L2     B7,B5,B5        ; |140|

           LDH     .D2T2   *+SP(40),B1     ; |164|
||         AND     .L2     3,B8,B7         ; |142|

           LDH     .D2T2   *+SP(46),B4     ; |145|
||         SHL     .S2     B7,5,B7         ; |142|
||         SHL     .S1X    B9,7,A6         ; |142|

           LDH     .D1T1   *+A7(14),A0     ; |181|
||         OR      .L1X    B7,A6,A6        ; |142|
||         EXTU    .S2     B6,26,27,B6     ; |142|
||         LDH     .D2T2   *+SP(78),B8     ; |145|

           OR      .L1X    B6,A6,A6        ; |142|
||         LDH     .D2T2   *+SP(80),B6     ; |145|

           NOP             3

           STW     .D2T1   A0,*+SP(176)    ; |181|
||         AND     .L2     7,B8,B8         ; |145|
||         SHL     .S1X    B1,7,A10        ; |164|
```

```
           SHL     .S1X    B4,7,A0           ; |145|
||         AND     .L2     7,B6,B1           ; |145|
||         SHL     .S2     B8,4,B6           ; |145|

           OR      .L1X    B6,A0,A1          ; |145|
||         LDH     .D1T1   *+A7(30),A0       ; |199|

           SHL     .S2     B1,1,B6           ; |145|
           NOP             3
           STW     .D2T1   A0,*+SP(192)      ; |199|
           LDH     .D2T1   *+SP(82),A3       ; |145|
           LDH     .D1T1   *+A7(20),A0       ; |189|
           LDH     .D2T2   *+SP(88),B10      ; |152|

           STB     .D1T2   B5,*A11++         ; |140|
||         LDH     .D2T1   *+SP(108),A14     ; |167|

           LDH     .D2T2   *+SP(82),B5       ; |149|

           EXTU    .S1     A3,29,31,A3       ; |145|
||         LDH     .D2T1   *+SP(98),A15      ; |155|

           STW     .D2T1   A0,*+SP(196)      ; |189|
||         MV      .L2X    A3,B13            ; |199|

           LDH     .D2T2   *+SP(92),B9       ; |155|
||         OR      .L1X    B6,A1,A3          ; |145|

           LDH     .D2T2   *+SP(94),B0       ; |155|
||         MV      .L1X    B13,A0            ; |189|
||         LDH     .D1T1   *+A7(4),A5        ; |177|

           OR      .L1     A0,A3,A3          ; |145|
||         STB     .D1T1   A6,*A11++         ; |142|
||         SHL     .S1X    B10,5,A8          ; |152|
||         SHL     .S2     B5,6,B10          ; |149|
||         LDH     .D2T2   *+SP(104),B5      ; |167|

           STB     .D1T1   A3,*A11++         ; |145|
||         LDH     .D2T2   *+SP(32),B4       ; |164|

           LDH     .D2T1   *+SP(48),A4       ; |167|
||         EXTU    .S1     A14,29,31,A3      ; |167|

           EXTU    .S1     A9,29,26,A14      ; |171|
||         LDH     .D1T1   *+A7(28),A9       ; |196|

           LDH     .D2T2   *+SP(96),B8       ; |155|
||         AND     .L2     7,B0,B13          ; |155|

           SHL     .S1X    B9,7,A13          ; |155|
```

```
||         SHL     .S2     B13,4,B5          ; |155|
||         AND     .L2     7,B5,B13          ; |167|
||         LDH     .D1T1   *A7,A12           ; |174|

           STW     .D2T1   A14,*+SP(212)     ; |196|
||         AND     .L2     3,B4,B0           ; |164|
||         SHL     .S1     A5,7,A14          ; |177|

           OR      .L1X    B5,A13,A5         ; |155|
||         STW     .D2T1   A8,*+SP(184)      ; |189|
||         SHL     .S1     A4,7,A1           ; |167|
||         SHL     .S2     B0,5,B5           ; |164|

           EXTU    .S1     A15,29,31,A0      ; |155|
||         STW     .D2T1   A9,*+SP(208)      ; |196|
||         OR      .L1X    B5,A10,A9         ; |164|
||         MV      .D1     A1,A4             ; |189|

           MV      .L1     A4,A15            ; |203|
||         STW     .D2T1   A0,*+SP(200)      ; |193|
||         AND     .L2     7,B8,B5           ; |155|
||         SHL     .S2     B13,4,B8          ; |167|

           OR      .L1X    B8,A15,A0         ; |167|
||         LDH     .D2T2   *+SP(84),B11      ; |149|
||         SHL     .S1     A12,5,A15         ; |174|
||         LDH     .D1T1   *+A7(36),A12      ; |203|

           STW     .D2T1   A3,*+SP(204)      ; |167|
           LDH     .D2T2   *+SP(90),B2       ; |152|
           LDW     .D2T1   *+SP(184),A3      ; |167|
           LDH     .D2T2   *+SP(92),B3       ; |152|
           STW     .D2T1   A12,*+SP(216)     ; |203|
           LDW     .D2T1   *+SP(156),A12     ; |203|
           AND     .L2     7,B2,B6           ; |152|
           SHL     .S2     B6,2,B4           ; |152|
           EXTU    .S2     B11,29,26,B9      ; |149|

           STW     .D2T1   A0,*+SP(184)      ; |203|
||         EXTU    .S2     B3,29,30,B11      ; |152|
||         OR      .L1X    B4,A3,A3          ; |152|

           OR      .L1X    B11,A3,A0         ; |152|
||         LDH     .D2T2   *+SP(86),B12      ; |149|
||         EXTU    .S1     A12,29,30,A3      ; |174|

           MV      .L2X    A3,B11            ; |189|
||         MV      .L1     A0,A3             ; |189|
||         LDW     .D2T1   *+SP(160),A0      ; |189|

           SHL     .S2     B5,1,B5           ; |155|
```

```
           LDW     .D2T1   *+SP(168),A4      ; |181|
           OR      .L2     B9,B10,B9         ; |149|
           AND     .S2     7,B12,B12         ; |149|

           EXTU    .S1     A0,29,26,A5       ; |181|
||         OR      .L1X    B5,A5,A0          ; |155|

           LDW     .D2T1   *+SP(200),A0      ; |181|
||         MV      .L1     A0,A5             ; |181|
||         MV      .L2X    A5,B12            ; |181|
||         OR      .S2     B12,B9,B9         ; |149|

           LDH     .D2T2   *+SP(98),B1       ; |159|
           LDH     .D1T1   *+A7(16),A12      ; |189|
           STB     .D1T2   B9,*A11++         ; |149|
           LDH     .D2T2   *+SP(108),B7      ; |171|

           SHL     .S1     A4,6,A0           ; |181|
||         OR      .L1     A0,A5,A3          ; |155|
||         STB     .D1T1   A3,*A11++         ; |152|
||         LDH     .D2T2   *+SP(48),B6       ; |164|

           STW     .D2T1   A0,*+SP(156)      ; |181|
           LDW     .D2T1   *+SP(212),A4      ; |181|
           LDW     .D2T1   *+SP(172),A0      ; |181|

           LDH     .D2T2   *+SP(100),B7      ; |159|
||         SHL     .S2     B7,6,B2           ; |171|
||         LDH     .D1T1   *+A7(26),A6       ; |196|

           LDH     .D2T2   *+SP(102),B3      ; |159|
           LDW     .D2T2   *+SP(164),B9      ; |181|

           STB     .D1T1   A3,*A11++         ; |155|
||         LDH     .D2T2   *+SP(24),B4       ; |162|

           LDH     .D2T2   *+SP(40),B13      ; |162|
||         EXTU    .S1     A0,29,31,A3       ; |177|
||         AND     .L1     7,A2,A10          ; |174|
||         EXTU    .S2     B6,26,27,B6       ; |164|

           EXTU    .S2     B7,29,26,B7       ; |159|
||         SHL     .S1     A10,2,A4          ; |174|
||         OR      .L1X    A4,B2,A0          ; |171|
||         STW     .D2T1   A6,*+SP(180)      ; |196|

           LDW     .D2T1   *+SP(180),A15     ; |181|
||         SHL     .S2     B1,6,B8           ; |159|
||         OR      .L1     A4,A15,A4         ; |174|
||         AND     .L2     7,B3,B5           ; |159|
```

```
           OR      .L2     B7,B8,B8          ; |159|
||         AND     .S2     7,B9,B9           ; |177|

           SHL     .S2     B9,4,B7           ; |177|
||         STW     .D2T1   A0,*+SP(200)      ; |177|
||         OR      .L2     B5,B8,B5          ; |159|

           EXTU    .S2     B13,30,31,B5      ; |162|
||         STB     .D1T2   B5,*A11++         ; |159|
||         OR      .L1X    B6,A9,A0          ; |164|

           LDH     .D2T2   *+SP(106),B0      ; |167|
||         SHL     .S2     B4,1,B7           ; |162|
||         OR      .L1X    B7,A14,A5         ; |177|
||         LDH     .D1T1   *+A7(8),A8        ; |177|

           LDH     .D1T1   *+A7(22),A15      ; |193|
||         LDH     .D2T2   *+SP(42),B7       ; |186|
||         OR      .L2     B5,B7,B5          ; |162|
||         SHL     .S1     A15,5,A9          ; |196|

           LDW     .D2T1   *+SP(184),A3      ; |196|
||         MV      .L2X    A3,B2             ; |177|

           LDW     .D2T1   *+SP(204),A4      ; |189|
||         MV      .L2X    A4,B13            ; |181|

           STB     .D1T2   B5,*A11++         ; |162|

           LDH     .D2T2   *+SP(112),B1      ; |171|
||         STB     .D1T1   A0,*A11++         ; |164|
||         AND     .L2     7,B0,B8           ; |167|

           SHL     .S2     B8,1,B6           ; |167|
||         LDW     .D2T1   *+SP(200),A0      ; |199|

           OR      .L1X    B6,A3,A3          ; |167|
||         STW     .D2T1   A8,*+SP(188)      ; |177|

           LDH     .D1T1   *+A7(18),A6       ; |189|
||         OR      .L1     A4,A3,A3          ; |167|

           LDW     .D2T2   *+SP(188),B6      ; |189|
||         STB     .D1T1   A3,*A11++         ; |167|

           LDW     .D2T1   *+SP(192),A4      ; |199|
||         AND     .L2     7,B1,B5           ; |171|

           LDW     .D2T1   *+SP(176),A14     ; |199|
||         OR      .L1X    B5,A0,A3          ; |171|
```

```
           LDH     .D2T2   *+SP(26),B9       ; |184|
||         STB     .D1T1   A3,*A11++         ; |171|

           LDW     .D2T1   *+SP(156),A3      ; |189|
||         AND     .L1     7,A6,A6           ; |189|

           AND     .L2     7,B6,B6           ; |177|
||         MV      .S2X    A6,B1             ; |189|

           SHL     .S2     B6,1,B5           ; |177|
||         STW     .D2T1   A9,*+SP(212)      ; |196|
||         MV      .L1X    B13,A4            ; |199|
||         SHL     .S1     A4,7,A6           ; |199|

           LDW     .D2T1   *+SP(196),A9      ; |199|
||         OR      .L2X    B5,A5,B5          ; |177|
||         MV      .L1X    B11,A0            ; |199|

           OR      .L1     A0,A4,A0          ; |174|
||         MV      .S1X    B12,A5            ; |189|

           OR      .L1     A5,A3,A0          ; |181|
||         STB     .D1T1   A0,*A11++         ; |174|
||         OR      .L2     B2,B5,B9          ; |177|
||         SHL     .S2     B9,1,B5           ; |184|
||         AND     .S1     7,A14,A14         ; |181|

           LDW     .D2T1   *+SP(208),A5      ; |193|
||         STB     .D1T2   B9,*A11++         ; |177|
||         OR      .L1     A14,A0,A0         ; |181|

           LDH     .D2T2   *+SP(34),B0       ; |186|
||         STB     .D1T1   A0,*A11++         ; |181|

           LDW     .D2T1   *+SP(212),A9      ; |203|
||         EXTU    .S1     A9,29,31,A4       ; |189|

           LDW     .D2T1   *+SP(216),A0      ; |203|
           LDH     .D1T1   *+A7(20),A1       ; |193|

           LDH     .D1T1   *+A7(32),A10      ; |199|
||         SHL     .S1X    B7,7,A5           ; |186|
||         AND     .L2X    7,A5,B7           ; |196|
||         LDH     .D2T2   *+SP(50),B3       ; |186|

           LDH     .D2T2   *+SP(42),B4       ; |184|
||         SHL     .S2     B7,2,B9           ; |196|

           LDH     .D1T1   *+A7(24),A13      ; |193|
||         LDH     .D2T2   *+SP(50),B10      ; |189|
||         AND     .L2     3,B0,B7           ; |186|
```

```
           SHL     .S1     A0,6,A12          ; |203|
||         AND     .L2X    7,A12,B0          ; |189|
||         SHL     .S2     B7,5,B9           ; |186|
||         OR      .L1X    B9,A9,A9          ; |196|
||         LDH     .D1T1   *+A7(34),A8       ; |199|

           SHL     .S2     B0,4,B0           ; |189|
||         MV      .L1X    B1,A0             ; |203|

           EXTU    .S2     B3,26,27,B6       ; |186|
||         OR      .L1X    B9,A5,A10         ; |186|
||         AND     .L2X    7,A10,B9          ; |199|
||         SHL     .S1     A1,6,A3           ; |193|
||         LDH     .D1T1   *+A7(38),A2       ; |203|

           EXTU    .S2     B4,30,31,B4       ; |184|
||         SHL     .S1     A0,1,A5           ; |189|
||         LDH     .D1T2   *+A7(30),B7       ; |196|

           SHL     .S2     B10,7,B8          ; |189|
||         LDH     .D1T1   *+A7(36),A0       ; |199|
||         AND     .L1     7,A13,A1          ; |193|

           OR      .L2     B0,B8,B4          ; |189|
||         OR      .S2     B4,B5,B5          ; |184|
||         LDH     .D1T1   *+A7(40),A10      ; |203|
||         OR      .L1X    B6,A10,A7         ; |186|
||         AND     .S1     7,A8,A8           ; |199|

           OR      .L1X    A5,B4,A13         ; |189|
||         SHL     .S1     A8,1,A8           ; |199|

           STB     .D1T2   B5,*A11++         ; |184|
||         SHL     .S2     B9,4,B4           ; |199|
||         EXTU    .S1     A15,29,26,A5      ; |193|

           OR      .L2X    B4,A6,B4          ; |199|
||         STB     .D1T1   A7,*A11++         ; |186|
||         OR      .L1     A4,A13,A4         ; |189|
||         OR      .S1     A5,A3,A3          ; |193|

           OR      .L1X    A8,B4,A4          ; |199|
||         OR      .S1     A1,A3,A3          ; |193|
||         STB     .D1T1   A4,*A11++         ; |189|

           STB     .D1T1   A3,*A11++         ; |193|
||         EXTU    .S2     B7,29,30,B4       ; |196|

           EXTU    .S1     A0,29,31,A0       ; |199|
||         OR      .L1X    B4,A9,A3          ; |196|
```

```
           LDW     .D2T2   *+SP(260),B12     ; |206|
||         OR      .L1     A0,A4,A3          ; |199|
||         STB     .D1T1   A3,*A11++         ; |196|
||         EXTU    .S1     A2,29,26,A2       ; |203|

           LDW     .D2T1   *+SP(232),A12     ; |206|
||         OR      .L1     A2,A12,A0         ; |203|

           LDW     .D2T2   *+SP(264),B13     ; |206|
           LDW     .D2T2   *+SP(256),B11     ; |206|

           LDW     .D2T2   *+SP(248),B3      ; |206|
||         STB     .D1T1   A3,*A11++         ; |199|

           LDW     .D2T1   *+SP(240),A14     ; |206|

           LDW     .D2T1   *+SP(244),A15     ; |206|
||         AND     .L2X    7,A10,B4          ; |203|

           LDW     .D2T1   *+SP(236),A13     ; |206|
||         OR      .L2X    B4,A0,B4          ; |203|

           LDW     .D2T1   *+SP(224),A10     ; |206|
||         STB     .D1T2   B4,*A11++         ; |203|

           B       .S2     B3                ; |206|
||         LDW     .D2T1   *+SP(228),A11     ; |206|

           LDW     .D2T2   *+SP(252),B10     ; |206|
           ADDK    .S2     264,SP            ; |206|
           NOP             3
           ; BRANCH OCCURS                   ; |206|
```

A.3 MPEG_Decode

Table A.3 shows the function profile for the MPEG_Decode benchmark. It lists the data for 10 functions where the application spends most of its execution time.

The columnns of the table can be described as follows:

% time	cumulative seconds	self seconds	calls	self us/call	total us/call	name
75.00	0.27	0.27	7920	34.09	34.09	Reference_IDCT
8.33	0.30	0.03	506880	0.06	0.06	putbyte
5.56	0.32	0.02	102121	0.20	0.20	Flush_Buffer
5.56	0.34	0.02	8238	2.43	2.43	form_component_prediction
5.56	0.36	0.02	12	1666.67	4166.67	store_yuv1
0.00	0.36	0.00	103514	0.00	0.00	Show_Bits
0.00	0.36	0.00	49847	0.00	0.20	Get_Bits
0.00	0.36	0.00	7920	0.00	0.00	Add_Block
0.00	0.36	0.00	7920	0.00	0.00	Clear_Block
0.00	0.36	0.00	4320	0.00	0.20	Get_Bits1

Table A.3: Function Profile for MPEG_Decode

- % time : the percentage of the total running time of the program used by this function.

- cumulative seconds : a running sum of the number of seconds accounted for by this function and those listed above it.

- self seconds : the number of seconds accounted for by this function alone. This is the major sort for this listing.

- calls : the number of times this function was invoked, if this function is profiled, else blank.

- self us/call : the average number of microseconds spent in this function per call, if this function is profiled, else blank.

- total us/call : the average number of milliseconds spent in this function and

its descendents per call, if this function is profiled, else blank.

- name : the name of the function. This is the minor sort for this listing. The index shows the location of the function in the gprof listing. If the index is in parenthesis it shows where it would appear in the gprof listing if it were to be printed.

The assembly language listing for the most frequently executed function is given below.

```
.sect ".text"
.global _Reference_IDCT

;******************************************************************************
;* FUNCTION NAME: _Reference_IDCT                                             *
;*                                                                            *
;*   Regs Modified     : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Regs Used         : A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,   *
;*                           A15,B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,   *
;*                           B13,SP                                           *
;*   Local Frame Size  : 0 Args + 532 Auto + 44 Save = 576 byte               *
;******************************************************************************
_Reference_IDCT:
;** --------------------------------------------------------------------------*
           ADDK    .S2     -576,SP           ; |76|
           STW     .D2T1   A13,*+SP(548)     ; |76|
           STW     .D2T2   B10,*+SP(564)     ; |76|
           STW     .D2T2   B12,*+SP(572)     ; |76|
           STW     .D2T2   B11,*+SP(568)     ; |76|
           STW     .D2T2   B3,*+SP(560)      ; |76|
           STW     .D2T1   A15,*+SP(556)     ; |76|
           STW     .D2T1   A14,*+SP(552)     ; |76|
           STW     .D2T1   A12,*+SP(544)     ; |76|
           STW     .D2T1   A11,*+SP(540)     ; |76|
           STW     .D2T1   A10,*+SP(536)     ; |76|
           STW     .D2T1   A4,*+SP(520)

           ADD     .L1X    8,SP,A13
||         MV      .L2X    A4,B10            ;
```

```
||         MVK     .S2     0x8,B12           ; |83|
||         STW     .D2T2   B13,*+SP(576)     ; |76|

           MVKL    .S1     _c,A14
           MVKH    .S1     _c,A14

           MV      .L1     A14,A10
||         MVK     .S2     0x8,B11           ; |84|

           MVK     .S2     0x8,B4            ; |88|

           MV      .L1X    B4,A15            ; |86|
||         ZERO    .S1     A12               ; |86|
||         ZERO    .D1     A11               ; |86|

;** --------------------------------------------------------------------------*
L4:
           B       .S1     __fltid           ; |89|
           LDH     .D2T1   *B10++,A4         ; |89|
           MVKL    .S2     RL64,B3           ; |89|
           MVKH    .S2     RL64,B3           ; |89|
           NOP             2
RL64:      ; CALL OCCURS                     ; |89|

           B       .S1     __mpyd            ; |89|
||         LDW     .D1T1   *A10,A4           ;
||         MV      .L2X    A4,B4             ;

           LDW     .D1T1   *+A10(4),A5       ;
||         MV      .L2X    A5,B5             ;

           MVKL    .S2     RL66,B3           ; |89|
           MVKH    .S2     RL66,B3           ; |89|
           NOP             2
RL66:      ; CALL OCCURS                     ; |89|
           B       .S1     __addd            ; |89|
           MV      .L2X    A5,B5             ;
           MVKL    .S2     RL68,B3           ; |89|
           MV      .S1     A12,A5            ; |89|
           MV      .L2X    A4,B4             ;

           MV      .L1     A11,A4            ; |89|
||         MVKH    .S2     RL68,B3           ; |89|

RL68:      ; CALL OCCURS                     ; |89|
           SUB     .L1     A15,1,A1          ;
   [ A1]   B       .S1     L4                ; |89|
           SUB     .L1     A15,1,A15         ;
           ADDK    .S1     64,A10            ; |89|
           MV      .L1     A4,A11            ;
           MV      .D1     A5,A12            ;
```

```
   [!A1]   SUB     .L1X    B11,1,A1          ;
           ; BRANCH OCCURS                   ; |89|
;** --------------------------------------------------------------------------*
   [ A1]   B       .S1     L4                ; |92|
           STW     .D1T1   A11,*A13++        ; |91|
           STW     .D1T1   A12,*A13++        ; |91|

           SUB     .L2     B11,1,B11         ;
||         ADDK    .S1     -504,A10
||         SUBAW   .D2     B10,4,B10
|| [ A1]   MVK     .S2     0x8,B4            ; |88|

   [ A1]   MV      .L1X    B4,A15            ; |86|
|| [ A1]   ZERO    .S1     A12               ; |86|
|| [ A1]   ZERO    .D1     A11               ; |86|

   [!A1]   SUB     .L1X    B12,1,A1          ;
           ; BRANCH OCCURS                   ; |92|
;** --------------------------------------------------------------------------*
   [ A1]   B       .S1     L4                ; |92|

           SUB     .L2     B12,1,B12         ;
||         ADDK    .S2     16,B10
|| [ A1]   MVKL    .S1     _c,A14

   [ A1]   MVKH    .S1     _c,A14
|| [!A1]   MVK     .S2     0x8,B4            ; |97|

   [ A1]   MV      .L1     A14,A10
|| [ A1]   MVK     .S2     0x8,B11           ; |84|

   [ A1]   MVK     .S2     0x8,B4            ; |88|

   [ A1]   MV      .L1X    B4,A15            ; |86|
|| [ A1]   ZERO    .S1     A12               ; |86|
|| [ A1]   ZERO    .D1     A11               ; |86|

           ; BRANCH OCCURS                   ; |92|
;** --------------------------------------------------------------------------*

           STW     .D2T2   B4,*+SP(524)      ; |97|
||         ZERO    .L1     A15               ;
||         MVK     .S2     0xffffff00,B13
||         ZERO    .L2     B12               ;

           LDW     .D2T2   *+SP(520),B4
           NOP             2
;** --------------------------------------------------------------------------*
L5:
           ZERO    .L1     A14
           MVKL    .S1     _c,A13
```

```
           ADD     .L2X    A15,B4,B11
||         MVK     .S2     0x8,B4            ; |98|

           STW     .D2T2   B4,*+SP(528)
           ADD     .L2     B12,SP,B4
;** --------------------------------------------------------------------------*
L6:

           MVK     .S2     0x8,B4            ; |102|
||         MVKH    .S1     _c,A13
||         ADD     .L2     8,B4,B10

           STW     .D2T2   B4,*+SP(532)
||         ADD     .L1     A13,A14,A10
||         ZERO    .S1     A12               ; |100|
||         ZERO    .D1     A11               ; |100|

;** --------------------------------------------------------------------------*
L7:

           B       .S1     __mpyd            ; |103|
||         LDW     .D2T2   *B10,B4           ;
||         LDW     .D1T1   *A10,A4           ;

           LDW     .D1T1   *+A10(4),A5       ;
||         LDW     .D2T2   *+B10(4),B5       ;

           MVKL    .S2     RL72,B3           ; |103|
           MVKH    .S2     RL72,B3           ; |103|
           NOP             2
RL72:      ; CALL OCCURS                     ; |103|
           B       .S1     __addd            ; |103|
           MV      .L2X    A4,B4             ;
           MVKL    .S2     RL74,B3           ; |103|
           MV      .S1     A12,A4            ; |103|
           MV      .L2X    A5,B5             ;

           MV      .L1     A11,A5            ; |103|
||         MVKH    .S2     RL74,B3           ; |103|

RL74:      ; CALL OCCURS                     ; |103|
           LDW     .D2T2   *+SP(532),B4      ; |103|
           NOP             4
           SUB     .L1X    B4,1,A1           ;
   [ A1]   B       .S1     L7                ; |103|
           SUB     .L2     B4,1,B4           ;
           STW     .D2T2   B4,*+SP(532)      ; |103|
           MV      .L1     A4,A12            ;
           MV      .D1     A5,A11            ;
```

```
           ADDK    .S2     64,B10            ; |103|
||         ADDK    .S1     64,A10            ; |103|

           ; BRANCH OCCURS                   ; |103|
;** --------------------------------------------------------------------------*
           B       .S1     __addd            ; |105|
           MVKL    .S2     RL78,B3           ; |105|
           MVKH    .S2     RL78,B3           ; |105|
           ZERO    .L2     B5                ; |105|
           MVKH    .S2     0x3fe00000,B5     ; |105|
           ZERO    .L2     B4                ; |105|
RL78:      ; CALL OCCURS                     ; |105|
           B       .S1     _floor            ; |105|
           MVKL    .S2     RL80,B3           ; |105|
           MVKH    .S2     RL80,B3           ; |105|
           NOP             3
RL80:      ; CALL OCCURS                     ; |105|
           B       .S1     __fixdi           ; |105|
           MVKL    .S2     RL82,B3           ; |105|
           MVKH    .S2     RL82,B3           ; |105|
           NOP             3
RL82:      ; CALL OCCURS                     ; |105|
           MVK     .S1     -256,A0           ; |106|
           CMPLT   .L1     A4,A0,A1          ;
   [ A1]   B       .S1     L9                ; |106|
   [ A1]   EXT     .S2     B13,16,16,B4      ; |106|

   [ A1]   MV      .L1X    B4,A0             ; |106|
|| [ A1]   LDW     .D2T2   *+SP(528),B4

           NOP             3
           ; BRANCH OCCURS                   ; |106|
;** --------------------------------------------------------------------------*
           B       .S1     L8                ; |106|
           MVK     .S1     256,A0            ; |106|
           CMPLT   .L1     A4,A0,A1          ; |106|
   [!A1]   MVK     .S2     0xff,B4           ; |106|

   [!A1]   MV      .L1X    B4,A0             ; |106|
|| [ A1]   EXT     .S1     A4,16,16,A0       ; |106|
||         LDW     .D2T2   *+SP(528),B4

           NOP             1
           ; BRANCH OCCURS                   ; |106|
;** --------------------------------------------------------------------------*
L8:
           NOP             2
;** --------------------------------------------------------------------------*
L9:
           NOP             1
           SUB     .L1X    B4,1,A1           ;
```

```
   [ A1]   B       .S1     L6                ; |107|
           MV      .L2X    A0,B5
           SUB     .L2     B4,1,B4           ;
           STH     .D2T2   B5,*B11++(16)     ; |106|

           STW     .D2T2   B4,*+SP(528)      ; |107|
||         ADD     .L1     8,A14,A14         ; |107|

   [ A1]   MVKL    .S1     _c,A13
|| [ A1]   ADD     .L2     B12,SP,B4
|| [!A1]   LDW     .D2T2   *+SP(524),B4

           ; BRANCH OCCURS                   ; |107|
;** --------------------------------------------------------------------------*
           ADD     .L1     2,A15,A15         ; |107|
           ADD     .L2     8,B12,B12         ; |107|
           NOP             2
           SUB     .L1X    B4,1,A1           ;
   [ A1]   B       .S1     L5                ; |107|
           SUB     .L2     B4,1,B4           ;
           STW     .D2T2   B4,*+SP(524)      ; |107|
   [ A1]   LDW     .D2T2   *+SP(520),B4
           NOP             2
           ; BRANCH OCCURS                   ; |107|
;** --------------------------------------------------------------------------*
           LDW     .D2T2   *+SP(560),B3      ; |108|
           LDW     .D2T2   *+SP(572),B12     ; |108|
           LDW     .D2T2   *+SP(568),B11     ; |108|
           LDW     .D2T2   *+SP(564),B10     ; |108|
           LDW     .D2T1   *+SP(556),A15     ; |108|
           LDW     .D2T1   *+SP(552),A14     ; |108|
           LDW     .D2T1   *+SP(548),A13     ; |108|
           LDW     .D2T1   *+SP(544),A12     ; |108|
           LDW     .D2T1   *+SP(540),A11     ; |108|

           B       .S2     B3                ; |108|
||         LDW     .D2T2   *+SP(576),B13     ; |108|

           LDW     .D2T1   *+SP(536),A10     ; |108|
           ADDK    .S2     576,SP            ; |108|
           NOP             3
           ; BRANCH OCCURS                   ; |108|
```

BIBLIOGRAPHY

[1] P. Faraboschi, G. Desoli, and J.A. Fisher. The Latest Word in Digital and Media Processing. *IEEE Signal Processing Magazine*, pages 59–85, Mar 1998.

[2] J.D. Mellot and F. Taylor. Very Long Instruction Word Architectures for Digital Signal Processing. In *Proceedings of the IEEE Conference on Acoustics Speech and Signal Processing*, pages 583–586, Apr 1997.

[3] J. Turley and H. Hakkarainen. TI's New 'C6x DSP Screams at 1,600 MIPS. *The Microprocessor Report*, 11:14–17, Feb 1997.

[4] D.M. Tullsen, S.J. Eggers, and H.M. Levy. Simultaneous Multithreading: Maximizing On-Chip Parallelism. In *Proceedings of the* 22^{nd} *Annual International Symposium on Computer Architecture*, pages 392–403, June 1995.

[5] D.M. Tullsen and S.J. Eggers and J.S. Emer and H.M. Levy and J.L. Lo and R.L. Stamm. Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor. In *Proceedings of*

the 23^{rd} *Annual International Symposium on Computer Architecture*, pages 191–202, May 1996.

[6] A. Bilas, J. Fritts, and J.P. Singh. Real-Time Parallel MPEG-2 Decoding in Software. In *Proceedings of the* 11^{th} *International Parallel Processing Symposium*, pages 197–203, Apr 1997.

[7] M. A. Bayoumi, editor. *Parallel Algorithms and Architectures for DSP Applications*. Kluwer Academic Publishers, 1991.

[8] S.W. Keckler and W.J. Dally. Processor Coupling: Integrating Compile Time and Runtime Scheduling for Parallelism. In *Proceedings of the* 19^{th} *Annual International Symposium on Computer Architecture*, pages 202–213, May 1992.

[9] S. Kaxiras, G. Narlikar, A.D. Berenbaum, and Z. Hu. Comparing Power Consumption of an SMT and a CMP DSP for Mobile Phone Workloads. In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 211–220, Dec 2001.

[10] E. Özer, T.M. Conte, and S. Sharma. Weld: A Multithreading Technique Towards Latency-tolerant VLIW Processors. In *Proceedings of the* 8^{th} *International Conference on High Performance Computing*, pages 192–203, Dec 2001.

[11] B.R. Rau. Dynamically Scheduled VLIW Processors. In *Proceedings of the* 26^{th} *International Symposium on Microarchitecture*, pages 80–90, Dec 1993.

[12] B.R. Rau and J.A. Fisher. Instruction-Level Parallel Processing: History, Overview and Perspective. *The Journal of Supercomputing*, 7(1):9–50, Jan 1993.

[13] Texas Instruments. *TMS320C6000 CPU and Instruction Set Reference Guide*, Jan 2000.

[14] T.E. Jeremiassen. A DSP with Caches - A Study of the GSM-EFR Codec on the TI C6211. In *Proceedings of IEEE International Conference on Computer Design*, pages 138–145, 1999.

[15] S. Srinivasan, V. Cuppu, and B. Jacob. Transparent Data-Memory Organisations for Digital Signal Processors. In *Proceedings of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'01)*, pages 44–48, 2001.

[16] C. Lee, M. Potkonjak, and W.H. Mangione-Smith. MediaBench: a tool for evaluating and synthesizing multimedia and communications systems. In *Proceedings of the* 30^{th} *Annual IEEE/ACM International Symposium on Microarchitecture*, pages 330–335, Dec 1997.

[17] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown. MiBench: A free, commercially representative embedded benchmark

suite. In *Proceedings of IEEE* 4^{th} *Annual Workshop on Workload Characterization*, pages 3–14, Dec 2001.

[18] C. Lee. *UTDSP Benchmark Suite*. University of Toronto, Canada, 1992.

[19] Texas Instruments. *TMS320C6000 Optimizing Compiler User's Guide*, Mar 2000.

[20] P. Kohout. Hardware For Real-Time Operating Systems. Master's thesis, University of Maryland (College Park), U.S.A., 2002.

[21] A.P. Chandrakasan, S. Sheng, and R.W. Broderson. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, 1992.

[22] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of the* 27^{th} *International Symposium on Computer Architecture*, pages 83–94, Jun 2000.

[23] M. Levy. C64x DSP Gets Performance Boost. *The Microprocessor Report*, 17:28–30, Apr 2003.

[24] R. Jain, C.J. Hughes, and S.V. Adve. Soft Real-Time Scheduling on Simultaneous Multithreaded Processors. In *Proceedings of the* 23^{rd} *IEEE International Real-Time Systems Symposium*, pages 134–145, Dec 2002.