Effects of NS32082 Memory Management Unit on Processor Through Put

National Semiconductor
Application Note 464
Chris Siegl
August 1986

INTRODUCTION

The purpose of this application note is to give a satisfactory answer to the question, ''How great is the performance penalty for using the NS32082 memory management unit?'' To arrive at a satisfactory answer a number of benchmarks have been run on the DB32000 board using the NS32032 with and without the NS32082 as well as the NS32016 with and without the NS32082. The benchmarks were compiled on two different compilers to show the differing effects of the MMU based on the degree of code optimization. The results are tabulated in a table along with the percent performance penalty.

The results show that the percentages vary over the wide range of 6% to 18.5% with generally a greater MMU impact with higher levels of code optimization in the compiler. The Whetstone benchmark has also been included to show the effects of the MMU on floating-point instructions. As can be seen in the tables the effects are much smaller with longer instructions such as the floating-point instructions. The last section of this ap-note rationalizes the differences in performance under varying conditions and gives some rules of thumb to use in applying this data to a specific case.

THE TEST SET-UP

To run this set of tests the DB32000 board was used. This board is a complete microprocessor system specifically designed to assist the user in evaluating and developing hardware and software for the NS32032 CPU, related slave processors (NS32081 FPU and NS32082 MMU) and support devices. Through the use of on board multiplexers the NS32016 and NS32008 CPU's can also be run on this board. The configuration of this board used for these tests consist of the NS32081 FPU (floating point unit), the NS32202 ICU (interrupt control unit), 256K of dynamic RAM, extensive ROM/EPROM capability, and two serial RS-232 ports as well as a parallel I/O port. See the DB32000 data sheet for more detailed information.

The TDS monitor (shipped installed on the DB32000 board) was then removed and replaced with MON32. This monitor is compatible with National's DBG16 debugger and allows downloading of code from a host computer through the debugger using an RS-232 link therefore allowing the host machine to be remote from the development environment. This can even be done over a modem line to the host.

A timing routine using the counters in the ICU was linked to the compiled benchmark programs before they were downloaded to the DB32000. A command to the debugger then started the timing program executing which in turn called the compiled benchmark after starting the ICU counters. After the benchmark completes, it returns to the timing routine where the counters are stopped and the execution time is read from the registers. This set-up and the timing program used are covered in detail in another application note titled ''Using the DB32000 Evaluation Board for Benchmarking''.

The SYS-32 Multi-User development system was used as the host. This system is based on the Series 32000 family, runs GENIX™ (National's version of Berkley 4.1 UNIX™) operating system in a demand paged virtual memory environment. The system supports up to eight simultaneous users, C and Pascal high level language compilers, a Series 32000 assembler, symbolic debugger and supports in-system emulation for the 32000 family. The minimum system configuration consists of 1.25 megabytes of RAM (expandable to 3.25 megabytes) 70 megabytes of hard disk (expandable to 490 megabytes) and a streamer tape drive for backup. For more detailed information on the SYS-32, please refer to the SYS-32 data sheet. The details of the DBG16 symbolic debugger's usage for down loading and execution of the benchmaks is covered in the ap-note ''Using the DB32000 Evaluation Board for Benchmarks''.

RESULTS

TABLES I, II and III show the results of running the benchmarks under the four different part combinations. As can be seen in tables the MMU penalty varies considerably from benchmark to benchmark and especially from one compiler to another. To set an understanding of why the variations are so big, we must look at how the 32000 family of CPU's operate in memory.

TABLE I
Benchmarks Executed on DB32000—All Processors Running at 10 MHz with no Wait States using Genix 4.1 C Compiler

Benchmark	NS32032 W MMU	NS32032 W/O MMU	MMU Penalty	NS32016 W MMU	NS32016 W/O MMU	MMU Penalty
Ackerman. c	4.72	4.32	9.3%	6.03	5.27	14.4%
BenchE. c	8.89	8.12	9.5%	11.97	10.50	14.0%
Puzzle. c	20.59	19.10	7.8%	26.96	23.65	14.0%
Sieve. c	19.42	18.09	7.4%	22.15	19.62	12.9%
Fibonacci. c	22.13	20.28	9.1%	26.31	23.61	11.4%
Longsearch. c	7.36	6.71	9.7%	10.31	8.70	18.5%

RRD-B30M105/Printed in U. S. A.

TABLE II
Benchmarks Executed on DB32000—All Processors Running at 10 MHz with no Wait States using Greenhill's C-32000 1.6.8 Compiler

Benchmark	NS32032 W MMU	NS32032 W/O MMU	MMU Penalty	NS32016 W MMU	NS32016 W/O MMU	MMU Penalty
Ackerman. c	3.75	3.30	13.6%	5.06	4.37	15.8%
BenchE. c	4.44	4.00	11.0%	4.76	4.48	6.3%
Puzzle. c	7.82	7.09	10.3%	9.61	8.57	12.1%
Sieve. c	17.71	16.41	7.9%	19.65	17.89	9.9%
Fibonacci. c	18.34	16.47	11.4%	24.87	21.17	17.5%
Longsearch. c	6.77	5.97	13.4%	8.75	7.48	17.0%

TABLE III
Benchmarks Executed on DB32000—All Processors Running at 10 MHz with no Wait States using Genix 4.1 Pascal Compiler

Benchmark	NS32032 W MMU	NS32032 W/O MMU	MMU Penalty	NS32016 W MMU	NS32016 W/O MMU	MMU Penalty
Whetstone. P	5.08	4.83	5.2%	6.17	5.63	9.6%

Both the NS32032 and the NS32016 have an eight byte queue for instruction prefetching. As a result of this queue having an MMU in the system has little effect on instruction fetching. An interesting test that helps in understanding this is to add wait states only to the code segment while using no waitstate RAM for the stacks and static data segments. These tests show a performance degradation of only 2 or 3% per waitstate. Another approach to demonstrating the same effect which is not dependent on a special hardware setup (controlling the number of wait states on different areas of memory space is done in hardware) is to generate a software loop which only uses the registers and immediate data for holding operands. A short example of such a program is shown in listing 1. Table IV shows the results obtained from timing this program both with and without the MMU. As can be seen from the times the penalty is very small, much less than 1%. This example clearly demonstrates that the queue is doing a good job of minimizing the effects of the MMU or waitstates on intruction fetching. This is why, even though the MMU lengthens each memory cycle by 25% (memory cycle goes from 4 t-states to 5) the net effect on performance is typically less than 10%. The penalty comes primarily from the lengthening of operand fetches. The NS32032 takes a much smaller penalty if the operands are primarily 32 bits or more in length. In that case the NS32032 is only doing half as many operand fetches as the NS32016, which has to do two accesses to get 32 bit operands. Another thing to note is that the performance times between NS32032 and the NS32016 is less than 1% in our software program loop test (see Table IV). This is because both processors are internally identical except in the queue and bus interface. If the queue keeps up and there are no stack or memory reference operations the execution time would be identical. The difference in time in this test is due to the queue not quite keeping up and the branch which purges the queue which the NS32032 reloads twice as fast.

TABLE IV
Benchmarks Executed on DB32000—All Processors Running at 10 MHz with No Wait States
(times are in microseconds)

Benchmark	NS32032 W MMU	NS32032 W/O MMU	MMU Penalty	NS32016 W MMU	NS32016 W/O MMU	MMU Penalty
Progloop.b.s	12622	12559	0.50%	12750	12668	0.65%
Progloop.w.s	13344	13291	0.40%	13432	13350	0.61%
Progloop.d.s	14988	14939	0.33%	15075	14992	0.55%

Tables I and II are the results of two different compilers using the same source files for input but generating code at different levels of optimization. The compiler in Table II optimizes to a much greater degree resulting in a much smaller ratio of instruction fetches to operand fetches while the table one compiler generates more code to do the same work. The number of operands does not decrease through optimization but extraneous code is eliminated, driving down the code to operand fetch ratio. As a result the penalty rises but is still in the neighborhood of 10%. The greater the complexity of the instruction the smaller the MMU penalty because the queue is more likely to keep up and a larger ratio of execution time to operands fetched especially with the NS32032. Table III gives the results of the Whetstone benchmark which illustrates this. The Whetstone benchmark is primarily floating point, the big NS32032 advantage comes from the operands being 32 or 64 bits in length. The NS32016 is making two times as many operand memory references as the NS32032 and therefore gets two times the MMU penalty.

CONCLUSIONS

After studying the above tests we can see the major factor effecting the performance penalty due to the MMU is the number of operand references and stack operations per unit of time. If operands are typically longer than 16 bits or the stack is heavily used, the NS32032 will show a much lower MMU penalty than the NS32016. However, even for the NS32016 the MMU penalty is seldom greater than 15% and typically half that for the NS32032. This penalty being so small makes a strong case for using the MMU even in systems not using a bulk memory device and benefiting from the page replacement aspects. The MMU can be useful in these non bulk memory applications for protection at the page level as well as for system debugging and program maintenance. If portions of the ROM based code require changes only the ROM holding the effected page table needs to be replaced with the new code being addable in any available ROM socket. The MMU with the on board breakpoint resistors and counter can often greatly simplify isolating bugs in the field where system disassembly on an ISE (In System Emulator) would be out of the question or inconvenient.

In bulk memory based systems there is no question that the performance improvements due to the MMU far outweigh the performance lost due to a longer memory cycle. For more details in this area see the technical note entitled ‘‘Series 32000 The Benefits of Demand Paged Virtual Memory’’.

LISTING 1

```
;#########################################################################
;       INLINE CODE LOOP
;       12-10-85 by Chris Siegl
;       all operands in registers
;#########################################################################
;       progloop.b.s = i's replaced by b at end of instructions - operands
;                      are bytes (8 bits)
;       progloop.w.s = i's replaced by w at end of instructions - operands
;                      are words (16 bits)
;       progloop.d.s = i's replaced by d at end of instructions - operands
;                      are double-words (32 bits)
        .program
-main::
        movi   0,r0     ;set loop counter to 0 for 256 loops
        movi   9,r3     ;put bcd values in r3 & r4
        movi   9,r4
        movi   r3,rl
        movi   r3,r2
        movi   r3,r5
        movi   r3,r6
```

```
loop:
        absi    r1,r2
        addi    r1,r2
        addci   r1,r2
        addpi   r3,r4
        subpi   r3,r4
        addqi   4,r1
        ashi    4,r1
        lshi    5,r1
        roti    6,r1
        andi    r2,r5
        comi    r2,r1
        ori     r2,r1
        xori    r2,r1
        nop
        muli    r5,r6
        absi    r1,r2
        addi    r1,r2
        addci   r1,r2
        addpi   r3,r4
        subpi   r3,r4
        addqi   4,r1
        ashi    4,r1
        lshi    5,r1
        roti    6,r1
        andi    r2,r5
        comi    r2,r1
        ori     r2,r1
        xori    r2,r1
        nop
        muli    r5,r6
        acbb    1,r0,loop
        rxp     0
        .endseg
```


National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408