Harris Semiconductor

APPNOTE

No. AN121.1 March 1997

Harris Digital

Harris 80C286 Performance Advantages Over the 80386SX

Author: Ted Dimbero

Introduction

The Harris 80C286, operating at the same frequency as the 80386SX, can outperform the 80386SX when executing software written for the 80286 and 8086, including all MS-DOS™, PC-DOS™ and OS/2™ -based programs. This performance advantage comes from the 80C286 requirement of fewer clock cycles to execute the same instructions as the 80386SX.

Industry standard 16-bit 8086/80286 code can execute 15-25% more efficiently on the 80C286 than on the 80386SX. There is no performance advantage gained by simply moving a system design from an 80C286 to an 80386SX. The 80C286 is the processor best suited for executing 16-bit 8086/80286 code.

The difference in clock cycle requirements is summarized in Figure 1. Of the 182 common instructions, the 80C286 executes 79 instructions faster than the 80386SX. Another 64 instructions execute in the same number of clock cycles on both processors.

Overall, 143 instructions (78% of all common instructions) execute as fast or faster on the 80C286 than on the 80386SX.

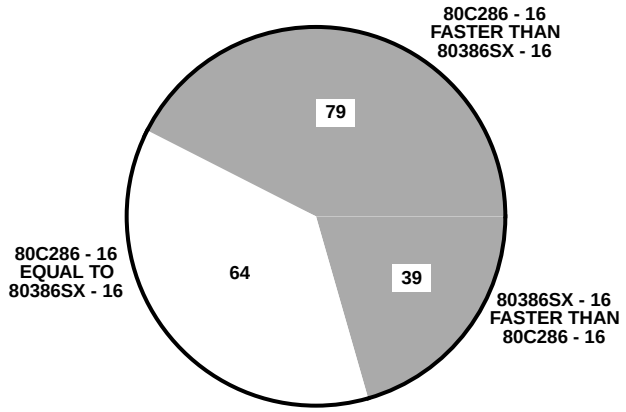


FIGURE 1. EXECUTION SPEED COMPARISON (NUMBER OF INSTRUCTIONS)

Taking this comparison one step further and looking at the performance contribution of each subset of instructions, those instructions that execute faster on the 80386SX are executed an average of 1.6 clock cycles faster. However, the instructions which execute faster on the 80C286 execute in an average of 23.1 clock cycles less than on the 80386SX.

Operating Speeds

The Harris 80C286 is available with operating frequencies of 12.5, 16, 20 and 25MHz. The 80386SX maximum operating frequency is limited to 16MHz. The instruction comparisons in Figure 1 (and throughout this document) evaluate the number of clock cycles required to execute the same instructions on both the 80386SX and the 80C286. Therefore, it illustrates the performance differences when the two processors are running at the same speed.

As this analysis shows, the 80C286 has the performance advantage when the two processors are running at the same speed. This 80C286 advantage is significantly greater when speed differences are taken into consideration.

The 80C286-25, for example, can execute 100% of the instructions available on both processors faster than a 16MHz 80386SX. In some cases, a 2X performance increase can be seen with the 80C286-25 compared to the 80386SX-16 (see Table 1).

This is not a benchmark number that can be credited to differences in system design; it illustrates a one-to-one comparison of the length of time it takes to execute the same instruction on the two processors. That is, given two equivalent systems (both having the same disk access speed, same number of wait states, similar cache controllers, etc), the 80C286-25 system would outperform the 80386SX system on any benchmark.

Table 1 illustrates a performance comparison of the 80386SX-16, 80C286-16, 80C286-20 and 80C286-25. The table lists the number of clock cycles needed to execute five different subroutines (see examples 1-5 in Subroutine Analysis section) and the total execution times for each subroutine for the two processors at varying operating frequencies.

In each of these examples, the results show that the 80C286 outperforms the 80386SX. In addition, the 80C286-25 shows performance increases of 170%-250% over the 80386SX-16. The 80C286 not only outperforms the 80386SX at similar operating frequencies, it also provides a path to increased performance by offering operating frequencies up to 25MHz which are not available with the 80386SX.

TABLE 1.

SUB-ROUTINES (PGS 6-9)	80386SX # CLOCK CYCLES	80C286 # CLOCK CYCLES	80386SX-16 EXECUTION TIME (μs)	80C286-16		80C286-20		80C286-25	
				EXECUTION TIME (μs)	PER-FORMANCE INCREASE OVER 80386SX-16	EXECUTION TIME (μs)	PER-FORMANCE INCREASE OVER 80386SX-16	EXECUTION TIME (μs)	PER-FORMANCE INCREASE OVER 80386SX-16
Example 1	104	73	6.5	4.6	141%	3.6	180%	2.92	223%
Example 2	2511	1899	156.9	118.7	132%	94.9	165%	75.9	132%
Example 3	2584	2380	161.5	148.7	108%	119.0	135%	95.2	170%
Example 4	837	583	52.3	36.4	143%	29.1	179%	23.3	224%
Example 5	307	193	19.2	12.1	158%	9.65	198%	7.7	249%

Architecture Background

The 80C286 static CMOS microprocessor combines low operating and standby power with high performance and operating frequencies up to 25MHz.

The 80C286 evolved from the industry standard 80C86 microprocessor and has vast architectural enhancements over its predecessor that allow the 80C286 to execute the same code with a significant performance increase.

Disregarding the clock speed increase, when upgrading from an 80C86 to an 80C286, the 80C286 can execute the same code with an increase in throughput of up to 4 times that of the 80C86. This increase is solely due to the architectural enhancements.

It is a common belief that replacing an 80C286 with the 80386SX microprocessor will yield similar performance increases. This is not the case. The new architecture gives the 80386SX 32-bit internal capability but it does not significantly increase the throughput of 16-bit 8086 or 80286 code.

When executing industry standard 8086 or 80286 code, replacing the 80C286 with an 80386SX does not result in a significant performance increase. In many cases, such a replacement will actually cause a performance degradation. This is evident in the following areas:

(1) Input/Output Handling

(2) Interrupt Handling

(3) Control Transfer (Loop, Jump, Call)

(4) 80286 Protected Mode Systems

(5) Multi-Tasking and Task Switching Operations.

The performance advantage of the 80C286 is especially evident in areas such as:

- Protected Mode Operating System - such as OS/2 and Xenix™
- Multi-Tasking Systems.
- Control Applications — utilizing interrupt and I/O instructions.
- Structured Software — utilizing many Control transfer instructions.
- Operating Systems that rely on interrupts to perform functions - such as MS-DOS, PC-DOS and OS/2.
- Upgrading 16-bit 80C86 applications for increased performance.

Figure 2 illustrates a comparison of the number of clock cycles needed to execute several instructions available on all three microprocessors (80C86, 80C286, and 80386SX). This illustrates the dramatic effect of 80C286 architectural enhancements on performance when compared to the 80C86 and the lack of similar performance improvement when executing 8086/80286 code on the 80386SX.

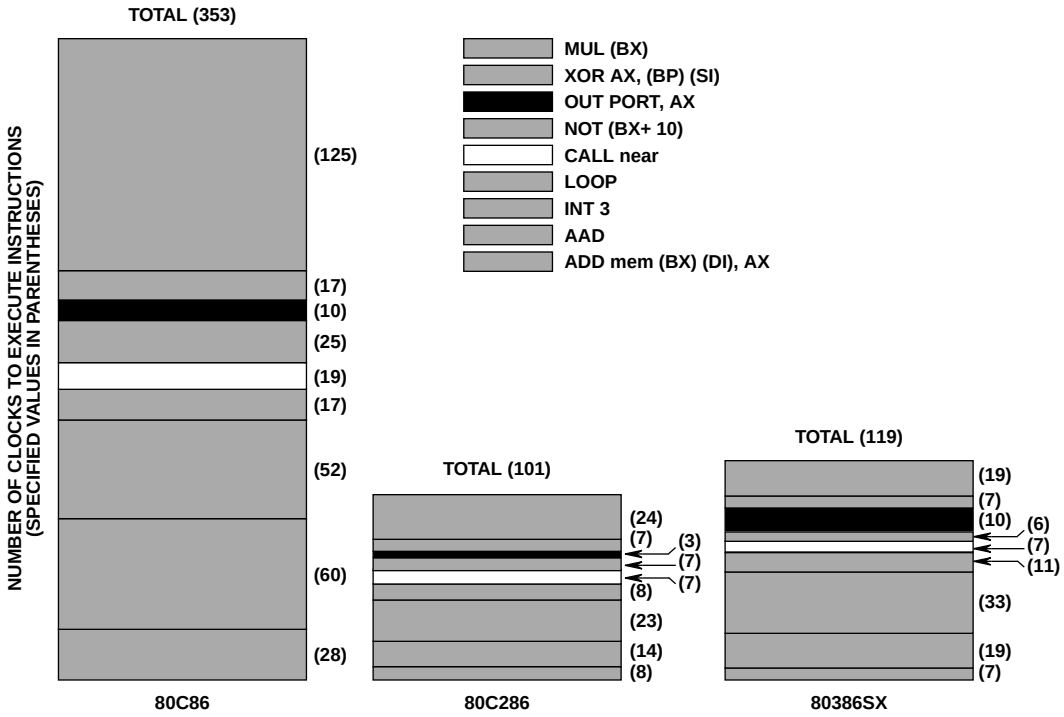


FIGURE 2. ARCHITECTURAL COMPARISON

Hardware System Comparison

Pipelined Operation on a 16-Bit Data Bus

At a given clock frequency, pipelined address operation increases a system's performance, while simultaneously allowing relatively slower memories and I/O devices to be used. Pipelined address operation provides the system increased address access time, and increased address decoding time.

80C286

The 80C286 supports a fully pipelined mode of operation for maximum system performance.

The 80C286 remains in a pipelined mode of operation even when idle bus cycles occur.

80386SX

The 80386SX does not support a fully pipelined mode of operation. Some pipelining can be achieved, but to accomplish this, external bus 'monitor' logic must be added to the system.

The 80386SX's pipelining is disrupted by idle bus cycles. A non-pipelined bus cycle, usually with an additional wait state, must be executed before the 80386SX can return to pipelined mode. Idle bus cycles occur an average of 9% of the time.

Idle Cycles

Another factor to consider when evaluating 80C286 and 80386SX performance is the effect of idle cycles on pipelined operation. Calculations have shown that, on average, bus idle cycles occur in the system approximately 9% of the time. The effect of idle cycles on pipelining is quite different on the 80C286 than on the 80386SX.

The 80C286 pipelined operation is not affected by idle cycles. When an idle cycle or cycles occur in a stream of pipelined bus cycles, the 80C286 returns to pipelining bus cycles immediately after the last idle cycle. In this way, each device on the bus (e.g. memory, peripheral) maintains a fixed timing associated with that device, and therefore always uses the minimum number of wait states required for that device.

On the other hand, the 80386SX pipelined operation is disrupted by idle cycles. With the 80386SX, an idle cycle or cycles occurring in a pipelined stream of bus cycles breaks the pipelining operation. Once an idle cycle has occurred, a non-pipelined bus cycle must always be executed prior to resuming pipelining. Since a non-pipelined bus cycle will have different timing than a pipelined bus cycle (even to the same device), an additional wait state must be added to this bus cycle. This not only degrades performance, but requires additional external logic to differentiate between a pipelined bus cycle access, and a non-pipelined bus cycle access, even to the same device with the same address.

From the preceding, it can be seen that when executing 16-bit code, the 80C286 has a 9% performance increase over the 80386SX, due to the manner in which each processor handles idle cycles alone.

Instruction Comparison

The Appendix in this document illustrates a direct comparison of the number of clock cycles needed to execute the same instructions on the 80C286 and the 80386SX. The table includes examples of instruction timing for all instructions available on both processors. Several addressing modes of each instruction type are included.

Of the 182 instruction examples analyzed, 79 of the instructions execute faster on the 80C286 than on the 80386SX; 64 of the instructions analyzed execute in the same number of clock cycles on both processors. This leaves only 39 instructions with improved performance on the 80386SX (See Figure 2). Over 78% of the instructions analyzed execute as fast or faster on the 80C286 than on the 80386SX.

This is vastly different than the previous 8086-to-80286 upgrade. With that upgrade, the 80C286 exhibits equal or better performance than the 80C86 with 100% of the instructions.

This clearly indicates that the 80C286 is the processor best suited for executing industry standard 8086 and 80286 code.

The following discussion groups each of the instructions into one of several categories to analyze which applications will benefit from utilizing the 80C286. The categories used are:

- Jumps, Calls, Returns and Loops (Real Mode).
- I/O Instructions.
- Logic, Arithmetic, Data Transfer, Shift and Rotate Instructions.
- Interrupts.
- Miscellaneous Instructions.
- Protected Mode/Multi-Tasking Instructions.

Jumps, Calls and Loops

In real mode, near calls, jumps, and conditional jumps (transfers within the current code segment) all take the same number of clock cycles to execute on the 80C286 and the 80386SX. Since the segment sizes are larger on the 80386SX, the near transfer instructions on the 80386SX can transfer a greater distance.

The far calls and jumps (transfers that switch to a new code segment; i.e., a code segment context switch) are faster on the 80C286: four clocks and one clock respectively. The far return instruction executes in three less clock cycles on the 80C286, and the near return takes one extra clock cycle. The protected mode calls, jumps, and returns are all faster on the 80C286 and are discussed in the section on Protected Mode.

The loop instruction is three clock cycles faster on the 80C286 than the 80386SX. Thus, the 80C286 would save 300 clock cycles over the 80386SX if a LOOP instruction were executed 100 times.

INSTRUCTION	ADVANTAGE		
	80C286	NONE	80386SX
Near JMP and Call		X	
Far CALL, JMP and RET	X		
LOOP	X		

I/O Instructions

The 80C286 has a significant advantage with the I/O instructions. The IN instruction is almost 2 1/2 times faster on the 80C286; the 80386SX takes 7 extra clock cycles to execute the same instruction. The OUT instruction is over 3 times faster on the 80C286; again the 80386SX takes 7 extra clock cycles to execute the same instruction. Executing the I/O instructions on the 80386SX is equivalent to executing on the 80C286 with 7 wait states.

The string I/O instructions (INS and OUTS) are also significantly faster on the 80C286. The INS instruction is 10 clock cycles faster on the 80C286, and the OUTS instruction is 9 clock cycles faster. This is particularly important if the string operations are going to be used to input or output a large block of data using the REP prefix. Inputting 100 words of data with the REP INS instruction is 208 clock cycles faster on the 80C286. An even more significant difference can be seen when outputting 100 words with the REP OUTS instruction. In this case, the 80C286 is 800 clock cycles faster than the 80386SX.

INSTRUCTION	ADVANTAGE		
	80C286	NONE	80386SX
IN	X		
OUT	X		
INS	X		
OUTS	X		

Logic, Arithmetic, Data Transfer, Shift and Rotate Instructions

Most forms of the logic, arithmetic, and data transfer instructions execute in the same number of clock cycles on both processors. Certain operand combinations of these instructions (immediate to register for example) take one extra clock cycle to execute on the 80C286.

In real mode, the segment register transfer instructions execute as fast or faster on the 80C286 than they do on the 80386SX. For example, using the POP instruction to transfer data into a segment register is 2 clock cycles faster on the 80C286.

Most of the string manipulation instructions execute in the same number of clock cycles on both processors. The MOVS and STOS instructions are faster on the 80C286.

The divide instruction executes in the same number of clock cycles on both processors. The number of clocks to execute the multiply instruction on the 80386SX is data dependent;

the number of clocks to execute the same instruction on the 80C286 is fixed. On average, the multiply instruction is five clock cycles faster on the 80386SX, but independing on the data, the 80386SX could be as many as 4 clock cycles slower than the 80C286.

The rotate and shift instructions are faster on the 80386SX. Unlike the 80C286, the 80386SX rotate and shift instructions do not depend on the number of bits to be shifted or rotated. Thus, the 80386SX has the advantage with multi-bit rotate and shift instructions. The 80C286 does, however, execute single bit rotate and shift instructions faster.

INSTRUCTION	ADVANTAGE		
	80C286	NONE	80386SX
Most Logic and Arithmetic		X	
Certain Operand Combinations of Logic and Arithmetic			X
Divide		X	
Multiply			X
Single Bit Shift or Rotate	X		
Multi-Bit Shift or Rotate			X
String Instructions	X		

Interrupt Instructions

Interrupts are serviced more quickly on the 80C286. The INT instruction, in real mode, executes 14 cycles faster on the 80C286 than it does on the 80386SX. The INTO, BOUND, and other instructions that can cause an interrupt all benefit from the faster interrupt handling features of the 80C286. The return from interrupt instruction (IRET) is 7 clock cycles faster on the 80C286. The PUSHA and POPA instructions, frequently used by interrupt handling procedures, are both faster on the 80C286. Protected Mode interrupt handling is discussed in the Protected Mode section.

INSTRUCTION	ADVANTAGE		
	80C286	NONE	80386SX
INT n	X		
INTO	X		
BOUND (If Interrupt)	X		
Break Point Interrupt	X		

Miscellaneous Instructions

The BCD instructions, HLT, and CBW execute from 1 to 5 clock cycles faster on the 80C286. The instructions to set and clear individual flags and the CWD instruction all execute in the same number of cycles on both processors. The ENTER, LEAVE, and BOUND instructions are from 1 to 3 cycles faster on the 80386SX. The BOUND instruction is only faster if an interrupt is not caused by the instruction.

INSTRUCTION	ADVANTAGE		
	80C286	NONE	80386SX
BCD Instructions	X		
Data Conversion (CBW, CWD)	X		
Flag Settling and Clearing		X	
BOUND (If No Interrupt)			X

Protected Mode/Multi-Tasking

When executing 80286 protected mode code, the 80C286 significantly out-performs the 80386SX. Task switching operations execute 100 to 271 clock cycles faster on the 80C286. The instruction to return from a called task is 63 clock cycles faster on the 80C286. This results in a very significant performance increase for systems utilizing the multi-tasking features. The 80C286 is clearly better suited than the 80386SX for running protected mode operating systems such as OS/2 and Xenix.

Inter-segment JMP, CALL and segment loading instructions also operate faster on the 80C286. The 80C286 saves anywhere from 4 to 11 clock cycles depending on the particular inter-segment transfer instruction. In protected mode, the inter-segment return is also faster on the 80C286. The 80C286 is 11 clock cycles faster when executing an inter-segment return to the same privilege level and is 17 cycles faster on inter-segment returns to a different privilege level.

The instructions to initialize and check the protected mode registers execute as fast or faster on the 80C286. The lDTR access instructions are an exception to this in that they take one extra clock cycle to execute on the 80C286. The instruction to switch the processor to protected mode (LMSW) is 7 cycles faster on the 80C286.

Most of the 80286 protected mode access checking instructions operate as fast or faster on the 80C286 than on the 80386SX. The LAR instruction is one clock cycle faster on the 80C286 and the LSL instruction is 5 clock cycles faster. The VERW instruction executes in the same speed on both processors and the VERR is 5 cycles faster on the 80386SX. The ARPL instruction used in protected mode procedures for pointer validation is 10 clock cycles faster on the 80C286.

INSTRUCTION	ADVANTAGE		
	80C286	NONE	80386SX
Task Switching	X		
Segment Register Loading	X		
Inter-Segment Transfer	X		
System Register Instructions		X	
Inter-Segment Transfers	X		
Access Checking Instructions		X	

Subroutine Analysis

This section lists several subroutines and then compares the number of clock cycles each subroutine will take to execute on the 80C286 and on the 80386SX.

EXAMPLE 1

This interrupt routine outputs a character to a terminal via a UART. The AL register must contain the character to be output. The routine first checks the status of the UART to determine if it is busy. If it is busy, the routine loops until the UART is free; when the UART is free, the character is output. Following is a listing of the code and the clock clyde analysis for the OUT_CHAR routine.

This sample procedure executes about 25% faster on the 80C286 than on the 80386SX. The advantage is realized through the 80C286's faster interrupt handling and faster I/O instructions.

80C286 CLOCK CYCLES	80386SX CLOCK CYCLES	OUT_CHARACTER PROC NEAR		
3	4		PUSHF	; save caller's flags.
3	2		PUSH AX	; save data to be output.
5	12	CK_STATUS:	IN AL, PORT_STATUS	; Input UART status.
6	5		CMP AL, BUSY	; Check If UART Busy.
3/7	3/7		JE CK_STATUS	; If busy go check again.
5	4		POP AX	; If not busy restore AX
3	10		OUT OUT_PORT, AL	; and output data.
5	5		POPF	; Restore Flags
17	22		IRET	; Return.
23	37		INT x	; Instruction to initiate OUTCHAR
				; Interrupt.
73	104	Total cycles if UART not busy.		
18	24	Number of cycles added for each loop while UART is busy.		

EXAMPLE 1.

EXAMPLE 2

The second example outputs an entire string of characters using the previous interrupt routine (denoted by "INT x" in the code below). The DS:SI registers point to the beginning of the string to be output. The string is variable in length and must be terminated with the "$" character.

80C286 CLOCK CYCLES	80386SX CLOCK CYCLES	OUT_STRING PROC FAR		
17	18		PUSHA	; save caller's registers.
5	5	NEXT:	LODSB	; Load first char to be output.
3	2		CMP AL, "$"	; Check to see if End of string.
3/7	3/7		JE done	; If end then go to DONE.
73	104		INT x	; If not end output character.
7	7		JMP next	; Go get next char to output.
19	24	DONE:	POPA	; Restore Registers when done.
15	18		RET	; Far Return.
13	17		Call OUT_STRING	; Far Call to initiate.
				; OUT_STRING procedure.
79+91/char	91+121/char	Total number of clocks to start and end routine. +Number of additional clocks to output each character in the output string.		

EXAMPLE 2.

To output a string of 20 characters, the 80C286 would take 1,899 clock cycles; using the same routine, the 80386SX would take 2,511 cycles. Each time a string of 20 characters is output, the 80C286 will save 612 clock cycles; an 80C286 performance increase of almost 25%. The advantage is realized through the 80C286's faster interrupt handling, faster I/O instructions, faster FAR transfer instructions, and faster register saving and restoring instructions.

EXAMPLE 3

This example adds all the values of a source array in memory to the values of a destination array in memory. The result is stored in the destination array. Both arrays are assumed to be in the current data segment. The count (number of words in the array), offset of source array, and offset of destination array are all assumed to be placed on the stack (in that order) by the calling program. The source code for the procedure is listed below:

80C286 CLOCK CYCLES	80386SX CLOCK CYCLES	ADD_ARRAY PROC NEAR		
17	18		PUSHA	; save caller's registers.
2	2		MOV BP, SP	; Point BP to current stack
5	4		MOV CX, (bp+22)	; Load array size from stack
				; into CX.
5	4		MOV SI, (bp+20)	; Load offset of source array
				; from stack into SI.
5	4		MOV DI, (bp+18)	; Load offset of destination
				; array from stack into DI.
2	2		CLD	; Clear Direction Flag.
5	5	NEXT:	LODSW	; Load the source word into AX.
7	7		ADD (DI), AX	; Add source to destination.
3	2		ADD DI, 02	; Point DI to next data.
8/4	11		LOOP NEXT	; Continue to ADD all elements
				; in the two arrays.
19	24		POPA	; Restore Registers
11	10		RET 6	; Near return.
		; Following is the code necessary to set up and call the above procedure.		
5	5		PUSH count	; Put count parameter on stack
3	2		PUSH offset S_ARRAY	; Put offset of source array
				; on stack.
3	2		PUSH offset D_ARRAY	; Put offset of destination
				; array on stack.
7	7		CALL ADD_ARRAY	; Near Call to initiate
				; ADD_Array procedure.
84+(23*CX)-4	84+(25*CX)	Total number of clocks to start and end routine. +Number of additional clocks for each item in array to be added.		

EXAMPLE 3.

Both processors take the same number of clock cycles for initialization before the call and closing up after the call (84). The loop that does the adding is faster on the 80C286. To add two 100 word arrays, the 80C286 would take 2,380 clock cycles; the 80386SX takes 2,584 (an additional 204 clocks) to execute the same routine. In this example, the LOOP instruction gives the 80C286 the performance advantage over the 80386SX.

EXAMPLE 4

This procedure is an example of an operating system procedure developed for a protected mode multi-privilege level system. The procedure lNIT_SEGMENT is passed a segment selector on the stack and will load that entire segment with zero's. The procedure is designed to execute at privilege level zero with a call gate at privilege level 3; this allows procedures executing at any level to utilize the lNIT_SEGMENT procedure. INIT_SEGMENT provides protection checks to ensure that the procedure passing the parameter has valid access to the segment that it is trying to initialize. This prevents a procedure at privilege level three from initializing a segment at privilege level zero.

80C286 CLOCK CYCLES	80386SX CLOCK CYCLES	INIT_SEGMENT PROC FAR WC = 1		
17	18		PUSHA	; save caller's registers.
3	2		PUSH ES	; save ES register.
2	2		MOV BP, SP	; Point BP to top of stack.
5	4		MOV AX, (BP+22)	; Load AX with segment selector
				; passed as parameter on stack.
5	4		MOV BX, (BP+20)	; Load BX with return CS to
				; determine caller's CPL.
10	20		ARPL AX, BX	; Adjust the Privilege level of
				; the segment selector according
				; to the caller's CPL.
16	16		VERW AX	; Test for valid write access
3/7	3/7		JNE ERROR	; If no valid access go to error.
17	18		MOV ES, AX	; LOAD ES with segment to be
				; initialized.
14	20		LSL CX, AX	; Load segment size into CX.
2	2		XOR DI, DI	; Load zero into DI.
2	2		XOR AX, AX	; Load zero into AX.
2	2		CLD	; Clear decrement flag.
4+3*cx	5+5*cx		REP STOSB	; Init entire segment to 00.
2	2		CLC	; Clear carry to indicate segment
				; initialized with no errors.
20	21	DONE:	POP ES	; Restore ES register.
19	24		POPA	; Restore Register
55	72		RET 2	; Ret FAR to different privilege
2	2	ERROR:	STC	; SET carry to indicate error.
7	7		JMP DONE	
		; Code to push selector on stack and initiate INIT SEGMENT via call gate.		
3	2		PUSH DATA_SELECTOR	; Place Selector on stack.
82	98		CALL INIT_SEGMENT_GATE	; Instruction to initiate
				; INIT SEGMENT procedure.
253	299	Total clocks if ERROR because segment not accessible.		
283+(3*S)	377+(5*S)	Total number of clocks if segment is initialized to zeros. "S" represents size of segment in bytes.		

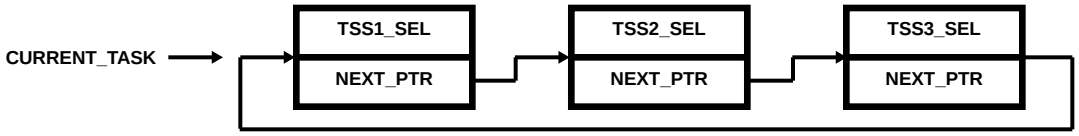
EXAMPLE 4.

This example shows that when executing instructions used for privilege verification and privilege level transitions the 80C286 is faster than the 80386SX. Without taking the LODS instruction into account, the 80C286 is 54 clock cycles faster when executing the same procedure. With the LODS instruction, and assuming a segment size of 100 bytes, the 80C286 would execute this routine 254 clock cycles faster than the 80386SX.

EXAMPLE 5

This Procedure is a task dispatcher that is invoked via an interrupt to cause a task switch to occur. This procedure utilizes a circular linked list of the tasks that need to be executed. A pointer called “CURRENT_TASK” points to the data structure for the current task being executed. The data structure contains the TSS for the task it is describing and a NEXT field that points to the data structure of the next task in the list to be executed. When the Task Dispatcher is invoked it switches the current pointer to the next task in the list and then invokes the new task by jumping to the TSS for that task. The data structure for the linked list is illustrated below.

The task dispatcher is actually a separate task that is invoked via an interrupt that signals that a new task should be initiated. Following is a listing for the simple task dispatcher.



80C286 CLOCK CYCLES	80386SX CLOCK CYCLES	TASK_DISPATCH PROC FAR		
5	4	START:	MOV BX, CURRENT TASK + 2	; Load BX with contents of next
				; field of current TASK. BX will
				; contain the address of the data
				; structure for next task to run
3	2		MOV CURRENT TASK, BX	; Update Current Task to point to
				; new task to be executed.
178	294		JMP DWORD PTR (BX-2)	; Start new task by jumping to TSS
7	7		JMP START	; JUMP to start for next time the
				; TASK dispatcher is invoked.
193	307			

EXAMPLE 5.

The advantage of the 80C286 in this case is in the faster task switch instruction. The task switch instruction is 116 clock cycles faster on the 80C286 than on the 80386SX. This performance increase makes the 80C286 the clear choice for multi-tasking applications.

Appendix

This appendix contains a table directly comparing the number of clock cycles necessary to execute all the instructions available on both the 80C286 and the 80386SX. The table includes several addressing modes of each instruction.

The table has five columns. The first column list the instruction being compared. The second column lists the number of clock cycles that the 80C286 needs to execute that instruction. The third column lists the number of clock cycles needed by the 80386SX to execute the same instruction. The fourth column divides the number of cycles needed by the 80386SX by the number of cycles needed by the 80C286. If this figure is greater than one, (see fifth column) then the 80C286 is faster than the 80386SX. For example, a 2.0 would indicate the 80C286 executes the same instruction twice as fast as the 80386SX. A 1.0 indicates that both processors execute the instruction in the same number of cycles. A number less than one indicates the 80386SX is faster than the 80C286.

APPENDIX TABLE

80C286 INSTRUCTION	NUMBER CLOCKS TO EXECUTE ON 80C286	NUMBER CLOCKS TO EXECUTE ON 80386SX	80386SX/ 80C286	80C286-16 FASTER THAN OR EQUAL TO 80386SX-16
AAA	3	4	1.33	√
AAD	14	19	1.36	√
AAM	16	17	1.06	√
AAS	3	4	1.33	√
ADC reg, reg	2	2	1.00	√
ADC mem, reg	7	7	1.00	√
ADC reg, immed	3	2	0.67	
ADC mem, immed	7	7	1.00	√
ADD reg, reg	2	2	1.00	√
ADD mem, reg	7	7	1.00	√
ADD reg, immed	3	2	0.67	
ADD mem, immed	7	7	1.00	√
AND reg, reg	2	2	1.00	√
AND mem, reg	7	7	1.00	√
AND reg, immed	3	2	0.67	
AND mem, immed	7	7	1.00	√
ARPL reg, reg	10	20	2.00	√
ARPL mem, reg	11	21	1.91	√
BOUND (no interrupt)	13	10	0.77	
CALL immed (near)	7	7	1.00	√
CALL immed (far real mode)	13	17	1.31	√
CALL immed (far PVAM)	26	42	1.61	√
CALL gate (same privilege PVAM)	41	64	1.56	√
CALL gate (different privilege PVAM)	82	98	1.19	√
CALL TSS (Task Switch PVAM)	177	285	1.61	√
CALL task_gate (Task Switch PVAM)	182	297	1.63	√
CBW	2	3	1.50	√
CLC	2	2	1.00	√
CLD	2	2	1.00	√
CLI	3	8	2.67	√
CLTS	2	5	2.50	√
CMC	2	2	1.00	√
CMP reg, reg	2	2	1.00	√
CMP mem, reg	6	5	0.83	

APPENDIX TABLE (Continued)

80C286 INSTRUCTION	NUMBER CLOCKS TO EXECUTE ON 80C286	NUMBER CLOCKS TO EXECUTE ON 80386SX	80386SX/ 80C286	80C286-16 FASTER THAN OR EQUAL TO 80386SX-16
CMP reg, immed	3	2	0.67	
CMP mem, immed	6	5	0.83	
CMPS	8	10	1.25	√
CWD	2	2	1.00	√
DAA	3	4	1.33	√
DAS	3	4	1.33	√
DEC reg	2	2	1.00	√
DEC mem	7	6	0.86	
DIV word, reg	22	22	1.00	√
DIV word, mem	25	25	1.00	√
ENTER immed1, immed2 (immed 2 = 6)	36	57	1.58	√
HLT	2	5	2.50	√
IDIV word, reg	25	27	1.08	√
IMUL word, mem	24	19	0.79	
IN	5	12	2.40	√
INC reg	2	2	1.00	√
INC mem	7	6	0.86	
INS	5	15	3.00	√
INT 3 (real mode)	23	33	1.43	√
INT immed (real mode)	23	37	1.61	√
INT immed (PVAM same privilege)	40	71	1.77	√
INT immed (PVAM different privilege)	78	111	1.42	√
INT TASK_GATE (PVAM Task Switch)	167	438	2.62	√
INTO (No Jump)	3	3	1.00	√
INTO (YES Jump real mode)	24	35	1.46	√
IRET (real mode)	17	24	1.41	√
IRET (PVAM same privilege)	31	42	1.35	√
IRET (PVAM different privilege)	55	86	1.56	√
IRET (PVAM task switch)	169	285	1.68	√
Jcond label (No jump)	3	3	1.00	√
Jcond label (Yes jump)	7	7	1.00	√
JMP near_label	7	7	1.00	√
JMP Far_label (real mode)	11	12	1.09	√
JMP FAR_LABEL (PVAM)	23	31	1.34	√
JMP CALL_GATE (PVAM same privilege)	38	53	1.35	√
JMP TASK_GATE (PVAM task switch)	183	298	1.63	√
JMP TSS (PVAM task switch)	178	289	1.62	√
LAHF	2	2	1.00	√
LAR reg	14	15	1.07	√
LAR mem	16	16	1.00	√
LDS (real mode)	7	7	1.00	√
LDS (PVAM)	21	23	1.33	√
LEA	3	2	0.67	
LEAVE	5	4	0.80	

APPENDIX TABLE (Continued)

80C286 INSTRUCTION	NUMBER CLOCKS TO EXECUTE ON 80C286	NUMBER CLOCKS TO EXECUTE ON 80386SX	80386SX/ 80C286	80C286-16 FASTER THAN OR EQUAL TO 80386SX-16
LGDT	11	11	1.00	√
LIDT	12	11	0.92	
LLDT reg	17	20	1.18	√
LLDT mem	19	24	1.26	√
LMSW reg	3	10	3.33	√
LMSW mem	6	13	2.17	√
LODS	5	5	1.00	√
LOOP (Jump)	8	11	1.38	√
LOOP (No Jump)	4	11	2.75	√
LSL reg	14	20	1.43	√
LSL mem	16	21	1.31	√
LTR reg	17	23	1.35	√
LTR mem	19	27	1.42	√
MOV reg, reg	2	2	1.00	√
MOV mem, reg	3	2	0.67	
MOV reg, immed	2	2	1.00	√
MOV mem, immed	3	2	0.67	
MOV seg_reg, reg (real mode)	2	2	1.00	√
MOV seg_reg, mem (real mode)	5	5	1.00	√
MOV seg_reg, reg (PVAM)	17	22	1.29	√
MOV seg_reg, mem (PVAM)	19	23	1.21	√
MOVS	5	7	1.40	√
MUL reg	21	15	0.71	
NEG reg	2	2	1.00	√
NEG mem	7	6	0.86	
NOP	3	3	1.00	√
NOT reg	2	2	1.00	√
NOT mem	7	6	0.86	
OR reg, reg	2	2	1.00	√
OR mem, reg	7	6	0.86	
OR reg, immed	3	2	0.67	
OR mem, immed	7	7	1.00	√
OUT	3	10	3.33	√
OUTS	5	14	2.80	√
POP reg	5	5	1.00	√
POP mem	5	7	1.40	√
POP seg_reg (real mode)	5	7	1.40	√
POP seg_reg (PVAM)	20	25	1.25	√
POPA	19	24	1.26	√
POPF	5	5	1.00	√
PUSH reg	3	2	0.67	
PUSH mem	5	7	1.40	√
PUSH seg_reg	3	2	0.67	
PUSHA	17	18	1.06	√

APPENDIX TABLE (Continued)

80C286 INSTRUCTION	NUMBER CLOCKS TO EXECUTE ON 80C286	NUMBER CLOCKS TO EXECUTE ON 80386SX	80386SX/ 80C286	80C286-16 FASTER THAN OR EQUAL TO 80386SX-16
PUSHF	3	4	1.33	√
RCR or RCL reg, 1	2	9	4.50	√
RCR or RCL mem, 1	7	10	1.43	√
RCR or RCL reg, cl (cl = 4)	9	9	1.00	√
RCR or RCL mem, cl (cl = 4)	12	10	0.83	
RCR or RCL reg, 4	9	9	1.00	√
RCR or RCL mem, 4	12	10	0.83	
ROR or ROL reg, 1	2	3	1.50	√
ROR or ROL mem, 1	7	7	1.00	√
REP INS (cx = 100)	405	613	1.51	√
REP MOVS (cx = 100)	405	407	1.00	√
REP OUTS (cx = 100)	405	512	1.26	√
REP STOS (cx = 100)	304	505	1.66	√
REP CMPS (cx = 100)	905	905	1.00	√
REPE CMPS (N = 100)	905	905	1.00	√
REPE SCAS (N = 100)	805	805	1.00	√
RET (near)	11	12	1.09	√
RET (far real mode)	15	20	1.33	√
RET (far PVAM same privilege)	25	36	1.44	√
RET (far PVAM different privilege)	55	72	1.31	√
SAHF	2	3	1.50	√
SHIFT reg, 1 (SHIFT = SAL, SAR, SHR)	2	3	1.50	√
SHIFT mem, 1	7	7	1.00	√
SBB reg, reg	2	2	1.00	√
SBB mem, reg	7	6	0.86	
SBB reg, immed	3	2	0.67	
SBB mem, immed	7	7	1.00	√
SCAS	7	7	1.00	√
SGDT	11	9	0.82	
SIDT	12	9	0.75	
SLDT reg	2	2	1.00	√
SLDT mem	3	2	0.67	
SMSW reg	2	2	1.00	√
SMSW mem	3	2	0.67	
STC	2	2	1.00	√
STD	2	2	1.00	√
STI	2	3	1.50	√
STOS	3	4	1.33	√
STR reg	2	22	1.00	√
STR mem	3	2	0.67	
SUB reg, reg	2	2	1.00	√
SUB mem, reg	7	7	1.00	√
SUB reg, immed	3	2	0.67	
SUB mem, immed	7	7	1.00	√

APPENDIX TABLE (Continued)

80C286 INSTRUCTION	NUMBER CLOCKS TO EXECUTE ON 80C286	NUMBER CLOCKS TO EXECUTE ON 80386SX	80386SX/ 80C286	80C286-16 FASTER THAN OR EQUAL TO 80386SX-16
TEST reg, reg	2	2	1.00	√
TEST mem, reg	6	5	0.83	
TEST reg, immed	3	2	0.67	
TEST mem, immed	6	5	0.83	
VERR reg	14	10	0.71	
VERR mem	16	11	0.69	
VERW reg	14	15	1.07	√
VERW mem	16	16	1.00	√
WAIT	3	6	2.00	√
XCHG reg, reg	3	3	1.00	√
XCHG reg, mem	5	5	1.00	√
XLAT	5	5	1.00	√
XOR reg, reg	2	2	1.00	√
XOR mem, reg	7	6	0.86	
XOR reg, immed	3	2	0.67	
XOR mem, immed	7	7	1.00	√
TOTAL number clocks to execute all instructions	6894	8664		
AVERAGE			1.25	
Number of Instructions faster on 80C286		79		
Number of Instructions equal on both processors		64		
Number of Instructions faster on 80386		39		
Total Number of instructions analyzed		182		