80386-CØ
STEPPING INFORMATION

CONFIDENTIAL

REVISION: Beta Site Version
3/30/87

This document contains specification changes and design notes.

Specification changes listed are permanent; the 80386 data sheet will be modified to incorporate the changes.

NOTES:

80386-B1 component identifier readable in DH after reset: 03H
80386-B1 revision identifier readable in DL after reset: 03H

CØ step { DH ——— C3
DL ——— C4

At this time, B1 stepping parts are identified with one of the marks shown below:

ii

ii A80386-16
ii S40344
ii (FPO number)
ii m c i '85 '86

ii

ii A80386-20
ii S40362
ii (FPO number)
ii m c i '85 '86

ii

ii A80386 ES B1
ii
ii
ii m c i '85 '86

Specification Changes

The specification changes numbered 1 through 4 for previous versions of the 80386 have now been incorporated in the latest version of the 80386 datasheet, version -002. The remaining specification changes, here, are now renumbered beginning with 1.

1. NT Bit and IOPL Bits in Real Mode

   The NT bit and IOPL bits of the FLAGS register can be set in Real Mode of the 80386. The exact behavior of these bits in 80386 Real Mode was not previously documented. Note that in 80286 Real Mode, these bits can not be set (they always remain 0 in 80286 Real Mode).

2. Coprocessor Data Pointer Stored by FSAVE/FSTENV Instructions is Undefined after Non-memory Instructions

   The contents of the operand address field resulting from a FSTENV or FSAVE are undefined if the preceeding coprocessor arithmetic instruction did not have a memory operand. The exact contents of the operand address field in this case was specified previously. This now confirms that the operand address field is undefined in that case.

3. Bit String Insert and Extract Instructions Removed

   Since the 80386 has unique and powerful 64-bit Double Shift instructions, and fast multi-bit shift and rotate instructions, the "Bit String Insert" and "Bit String Extract" instructions were removed. The insert/extract complex instructions did not provide an additional benefit that fully justified including them in 80386 silicon and all future compatible processors. A review concluded that the 80386 user obtains full performance in bit string manipulations using other powerful instructions such as 64-bit Double Shift, and other multi-bit shift/rotate instructions. These instructions support extremely fast manipulation of general unaligned bit strings of any length, by processing them in 32-bit chunks.

4. ERROR# Input Difference - Effect on PC/AT Compatible Coprocessor Connection

   On the 80386, latching the level of BUSY# when ERROR# becomes active will cause FST and FSTP instructions which get errors to hang the 80386. On the 80286, latching BUSY# when ERROR# becomes active (as performed in the PC/AT) did not cause any problems.

   Implications: The PC/AT uses a non-standard scheme to report 80287 errors to the 80286 (a scheme compatible with the non-standard scheme used to report 8087 errors to the 8088 in the original PC). The scheme used in the PC/AT works because a separate data channel is used by the 80286 to communicate with the 80287. However, the 80386 communicates with the math coprocessor using microcode loops. Therefore, PC/AT-compatible 80386 systems using an 80287 or 80387 numerics coprocessor must carefully follow the recommendation below when replicating the PC/AT's non-standard method of reporting coprocessor errors.

   How to properly replicate the PC/AT coprocessor error-reporting scheme: A workaround exists when replicating the PC/AT coprocessor interface in 80386-based systems. Note that this workaround needs to be incorporated for the non-standard PC/AT scheme; the standard recommended 80386/80387 connection functions properly and the 80386 implementation will not be altered. To understand the workaround, let us review the AT interface. In the PC/AT, the ERROR# input to the 80286 is tied inactive (high) permanently. The ERROR# output of the 80287 is tied to an interrupt port (IRQ13). This interrupt replaces error signalling via the 80286's ERROR# input. To guarantee (in the case of an 80287 error) that INTR 13 will be serviced prior to the execution of any further 80287 instructions, an

edge-triggered flip-flop latches BUSY# using ERROR# as a clock. The output of this latch is ORed with the BUSY# output of the 80287 and drives the BUSY# input of the 80286. This PC/AT scheme effectively delays BUSY# deactivation at the 80286 whenever an 80287 ERROR# is signalled. Since the 80286 BUSY# input remains active, the 80286 INTR 13 handler is guaranteed to execute before any other 80287 instructions may begin. The INTR 13 handler clears the BUSY# latch (via a write to a special I/O port) thus re-allowing execution of 80287 instructions. The INTR 13 handler then branches to the NMI handler, where the user-defined numerics error handler resides in PC-compatible systems.

The use of an interrupt guarantees that an error from a coprocessor instruction will be detected. Latching BUSY# guarantees that any coprocessor instruction (except FINIT, FSETPM, FCLEX) following the instruction that raised the error will not be executed before the NMI handler is executed. This approximates the way the 8087-8088 error-reporting interface works in the original PC.

The 80386 can use a PC/AT-compatible interface to communicate with an 80287/80387 provided that while BUSY# is latched active, the 80386 PEREQ input is also activated, and the 80287/80387 coprocessor is disabled. An 80287 can be disabled using either NPS1# or NPS2. An 80387 should be disabled using its STEN input (do not use the 80387 NPS1# or NPS2 inputs to disable the 80387 in this case). Note that while PEREQ is artificially activated as described above, the 80386 may issue I/O read cycles for the coprocessor. It is permissible for the 80386 data pins to float throughout such I/O read cycles.

5. Read Cycles Require Valid Data Bus Levels

The 80386 requires that all data bus pins be at a valid logic state (high or low) at the end of each read cycle, when READY# is asserted. The system MUST be designed to meet this requirement. Therefore, do NOT allow any data lines to be floating when the read cycle completes. NOTE: The I/O read cycles just mentioned in the previous item, item 4, are free from this requirement.

Implications: If the device being read is a 32-bit device, such as a 32-bit memory, the system should present 32-bits of data to the 80386 even if not all of the 80386 byte enables are asserted.

If the device being read is a 16-bit or an 8-bit device, however, pullup resistors can be used to guarantee valid logic levels on the upper data lines, which otherwise would be floating. Note that bus cycles to 16-bit and 8-bit devices typically include several wait states, but always calculate the effects of R-C time constants to ensure the pullups will drive proper logic levels onto the bus within the time required.

6. I/O Permission Bitmap Must Reside Within TSS Offset 0FFFFh

The 80386 requires that the entire I/O permission bitmap (including the terminating byte of "0FFh"), which is part of an 80386 TSS, begin at an offset no larger than 0DFFFh. This guarantees the entire bitmap (up to 8 kilobytes + 1 terminator byte of 0FFh) will reside at TSS offsets of 0FFFFh or less. Therefore, the pointer within a 386 TSS called Bit_Map_Offset(15:0) must contain a value of 0DFFFh or less under all conditions, even when you intend the Bit_Map_Offset to point beyond the limit of the TSS itself.

7. BS16# Must Not Be Asserted During Pipelined Bus Cycles

In datasheet figures 5-16, 5-17, 5-19, and 5-22, the bus size 16 (BS16#) input is shown as "don't care" during T2P and T21 in pipelined bus cycles. This is incorrect. In these figures, BS16# should be high during states T2P and T21. That is, once address pipelining has been requested by asserting next address (NA#), BS16# must be negated for the remainder of the current bus cycle.

Implications: Don't assert BS16# if NA# has already been sampled asserted in the current bus cycle.

8. Double Page Faults Do Not Raise Double Fault Exception

Problem: If a second page fault occurs, while the processor is attempting to enter the service routine for the first, then the processor will invoke the page fault (exception 14) handler a second time, rather than the double fault (exception 8) handler. A subsequent fault, though, will lead to shutdown.

Workaround: No workaround is necessary in a working system.

ANOMALIES

(read or write)

- Verify access rights followed by JMP w/ 32-bit displacement and had a limit violation in the process, HOLD CAN hang the processor in a window. Restartable by any interrupt.

Design Notes

1. **Read Cycles Require Valid Data Bus Levels**

   Please refer to Specification Change 5 for important news on proper system design for 386 read cycles.

2. Use of ESP as a Base Register With CALL, PUSH, and POP Instructions

   This clarifies how ESP behaves with instructions that implicitly reference the stack and explicitly reference another location in memory using ESP as a base register.

Instruction	Explicit Memory Reference uses the ESP value...	ESP value used as base
CALL-indirect-thru-memory	before decrementing	old ESP
PUSH-from-memory	before decrementing	old ESP
POP-to-memory	after incrementing	new ESP

   This is consistent in that the CALL-indirect-thru-memory and the PUSH-from-memory both use the same ESP value.

   Furthermore, the relation between PUSH-from-memory and POP-to-memory is such that it allows the instruction sequence:

```
PUSH [ESP+n]
POP  [ESP+n]
```

   to have the desirable property of both instructions referencing the same memory location.

3. Use of Code Breaks to Debug 86/286 Operating Systems

   The RF bit in the EFLAGS register is cleared by a 16-bit IRET, making it difficult to use the on-chip debug registers to set code breakpoints to debug 16-bit operating systems. Data breakpoints work fine in all cases, and code breakpoints work fine as long as all interrupt handlers are 32-bits and return with 32-bit IRETs or task switches. In 16-bit environments, software debuggers should use the CC (single byte INT 3 instruction) to place software breakpoints in code.

4. Use of ESP in 16-bit Code with 32-bit Interrupt Handlers

   When a 32-bit IRET is used to return to another privilege level, and the old level uses a 4G stack (B=1), while the new level uses a 64k stack (B=0), then only the lower word of ESP is updated. The upper word remains unchanged. This is fine for pure 16-bit code, as well as pure 32-bit code. However, when 32-bit interrupt handlers are present, 16-bit code should avoid any dependence on the upper word of ESP. No changes are necessary in existing 16-bit code, since the only way to access ESP in USE16 segments is through the 32-bit address size prefix.