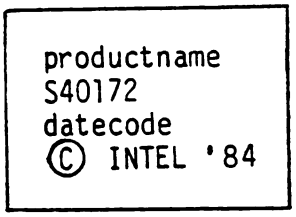
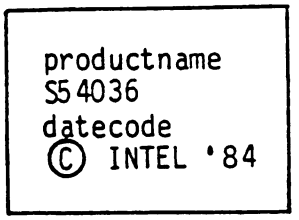
80286 (C-2) Information Package
21 November 1984

This applies to S-spec #40172 and to S-spec #54036 of the 80286, marked as



or



since they contain a (C-2) stepping of the 80286. Their complete parametric specifications are given by the datasheet in the 1984 Intel Microsystem Components Handbook, order number 230843.

Errata Items:
Below are full descriptions of any problem the 80286(C-2) is known to have, for your information when using this powerful processor. These items will be corrected in subsequent versions of the 80286:

1. DON'T REMOVE INTERRUPT SIGNAL EARLY
When the INTR is activated and external interrupts are enabled, INTR must be held active until the CPU begins performing the first INTA bus cycle to process the interrupt, or else unpredictable CPU behavior could result. See additional information page 6.

2. NESTED INTERRUPTS
When the 80286 is in Protected Mode, and processes an external interrupt or INT instruction which references an Interrupt Gate in the IDT, an external interrupt activating the INTR input will be processed if the INTR pin remains active or goes active anytime between the fifth and tenth bus cycles after the second INTA bus cycle for the first interrupt. Normally, if an external interrupt references an Interrupt Gate in the IDT, then no further interrupts are processed, until interrupts are reenabled within the service routine or by the IRET instruction at the end of the service routine. Because of this errata, an interrupt routine may be interrupted before its first instruction if the INTR input is active during the "window" described above.

3. NON-MASKABLE INTERRUPT
When INTR goes active, and then NMI goes active slightly later, exactly during the last internal clock period of the instruction prior to the INTR being processed, the NMI is not recognized. The probability of this occurring from unrelated, asynchronous INTR and NMI events is very unlikely. However, if the same event can activate INTR and NMI, ensure the NMI pin is activated 10ns or more before the INTR.

4. PROTECTION VIOLATIONS

The details of this are primarily of interest to an operating system writer.

The protection violations involved usually indicate a probable software bug and restart is not desired if one of these violations occurs. In a Protected Mode 80286 system with wait states during any bus cycles, when certain protection violations are detected by the 80286 component, and the component transfers control to the exception handling routine, the contents of the CX register may be unreliable. (Whether CX contents are changed is a function of bus activity at the time internal microcode detects the protection violation.)

Note that any "not present" exception when a CS, SS, DS or ES segment is "not present" is entirely restartable, for virtual memory implementation.

The protection violations which could lead to unreliable CX contents are below and note again these violations usually indicate a software bug Therefore restart is not usually desired after these protection violations**:

1) exception #GP(0) from attempted access to data segment or extra segment when the corresponding segment register holds a null selector,
2) exception #GP(0) from attempted data read from code segment when code segment has the "execute-only" attribute,
3) exception #GP(0) from attempted write to code segment (code segments are not writable), or to data segment or extra segment if the data or extra segment has the "read-only" attribute,
4) exception #NP(selector) from attempted load of a selector referencing the local descriptor table into CS, DS, or ES, when the LDT is not present (or exception #SS(selector) if SS),
5) exception #GP(0) from attempted input or output instruction when CPL IOPL,
6) exception #GP(selector) from attempted access to a descriptor in GDT, LDT, or IDT, beyond the defined limit of the descriptor table,
7) exception #GP(0) from attempted read or write (except for "PUSH" onto stack) beyond the defined limit of segment.

The following protection violation below may also lead to unreliable CX register contents. The following protection violation is designed to be restartable for dynamically growable stacks, but due to the errata, is not restartable on this stepping in a system which has wait states in any bus cycles (for example, refresh cycles in system with dynamic memory):

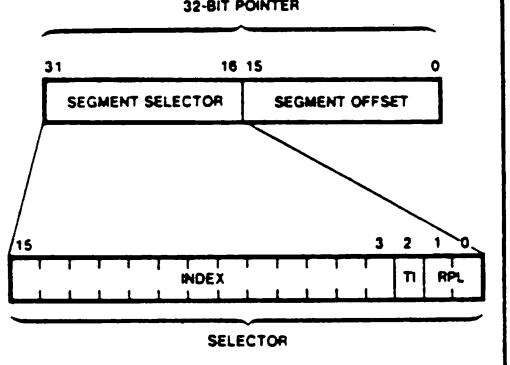
8) exception #SS(0) from attempted "PUSH" below the defined limit of the stack segment (restart allows dynamically growable stack segments).

** Notation...
exception #NP() = exception #11 = Not-Present Fault
exception #SS() = exception #12 = Stack Fault
exception #GP() = exception #13 = General Protection Fault
The value in parentheses indicates the type of error code pushed on exception handler's stack.

5. LOADING NULL SELECTOR VALUES INTO DS OR ES REGISTERS

This isn't usually a problem, since any of the 4 null selector values are equivalent in purpose. Just be aware of it: in Protected Mode, when any of the 4 null selector values (the 4 possible null selectors are 0000H, 0001H, 0002H and 0003H) are loaded into DS or ES registers via a MOV or POP instruction or a task switch, note the 80286 always loads the null selector 0000H into the corresponding registers. The 80286 will be improved to load all 4 null selector values literally. In Real Mode all values loaded into DS or ES are of course loaded unaltered.

Background: In Protected Mode, the null selector is any selector (see Figure below) whose Index bits and Table Indicator bit are all zero. Since the 2-bit RPL field may be 00, 01, 10 or 11 there are 4 possible null selectors. In hex format, the 4 possible null selectors are 0000H, 0001H, 0002H and 0003H. They all serve as "null" values, and a general protection violation correctly occurs if software attempts to access the data segment or extra segment when a null selector is in the corresponding segment register.



Protected Mode

Figure 6-1. Format of the Segment Selector Component

Documentation Corrections:

These items are documentation errors to be corrected as soon as possible. Accurate descriptions below are for your information.

1. SINGLE-STEPPING THE INT n INSTRUCTION

To prevent application software from invoking privileged system interrupt routines with the trap flag (single-step flag) set, the "INT n" instruction disables the Trap Flag if it was previously set. This applies to Real Address mode as well as Protected Mode, from the B-2 stepping onward. This prevents applications programs from invoking privileged interrupt routines with the TF set, causing single-stepping of operating system interrupt routines which may be time-critical, for example. The INT n instruction pushes the orginal flag word (before disabling TF), pushes the CS and IP pointing to the next instruction, then disables the Trap Flag before executing the interrupt routine. However, debuggers which single-step through code can still single-step within an interrupt routine called by the INT n instruction by recognizing the INT n opcode and emulating the INT n function (e.g. push Flags, CS, IP, and set CS:IP per the value found in the nth interrupt vector or IDT descriptor). Debuggers which emulate the INT n are backward compatible to earlier 80286 steppings, to the 80186, and 8086.

2. LOCK SIGNAL DURING INTA CYCLES

This does not affect the 8259A interrupt controller. From the (B-2) stepping onward, the 286 LOCK signal is active during both INTA cycles of the Interrupt Acknowledge sequence (on earlier steppings, the LOCK signal was active only on the first INTA cycle). Internal circuit design necessitated this correction. LOCK asserted means "lock this cycle TO THE NEXT bus cycle." Therefore the (B-2) and later steppings lock the two INTA cycles to the first stack push of the interrupt processing sequence. Doing so is no problem. However, custom interrupt controllers may be affected if they use the state of LOCK to distinguish the first and second INTA cycles. Custom hardware designers note you can deactivate the INTR signal to the 80286 anytime during either INTA cycle, and the vector may be placed on the data bus during both cycles although the 80286 only reads the vector during the second INTA cycle.

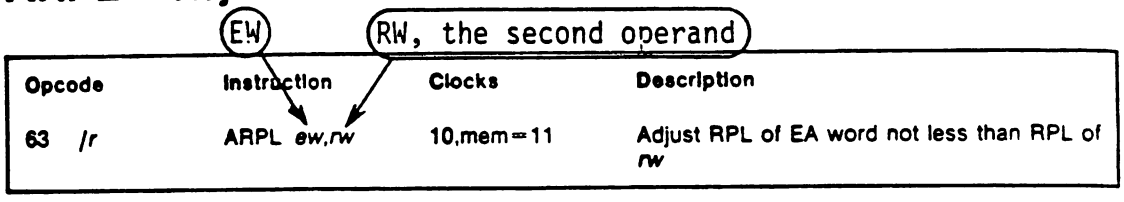
3. INSTRUCTIONS LONGER THAN 10 BYTES

Instructions longer than 10 bytes occur only by using the assembler to intentionally place multiple redundant prefixes (e.g. multiple lock prefixes and/or segment override prefixes) before valid opcode bytes. On all 80286 components, in Real Address Mode or Protected Mode, when the 80286 detects an instruction that is illegal solely due to being greater than 10 bytes in length, it generates an exception #13 (General Protection Exception) rather than exception #6 (Invalid Opcode) as previously described. Note that undefined opcodes do generate exception #6 (Invalid Opcode) as described.

4. ARPL INSTRUCTION

On any 80286 component, when the second operand of the ARPL instruction (see below) is a null selector, the instruction generates an exception #13. This is not a problem since the RW operand contains the RPL bits used only as a "standard" for comparison, and the EW operand is the actual selector whose RPL bits are subject to adjustment. Ensure the RW operand does not have all its bits 15-2 equal to "0", since that makes it a null selector value. This is easy to avoid since only RW bits 0 and 1 are used for comparison; bits 15-2 can be anything but all zeroes.

ARPL—Adjust RPL Field of Selector



Opcode	Instruction	Clocks	Description
63 /r	ARPL *ew,rw*	10,mem=11	Adjust RPL of EA word not less than RPL of *rw*

80286 Component Identification:

The C-2 stepping of the 80286 can be identified by the copyright marking of " © Intel '84".

The earlier B-2/B-3 stepping of the 80286 can be identified by the copyright marking of " © Intel '83".

The earlier B-1 stepping can be identified by the copyright marking of " © Intel '82".

The earlier A-1 stepping is identified by the marking "80286A1".

Parts marked with an "ES" are Engineering Samples which may not be fully production tested.

Background information on 80286 (C-2) errata #1:

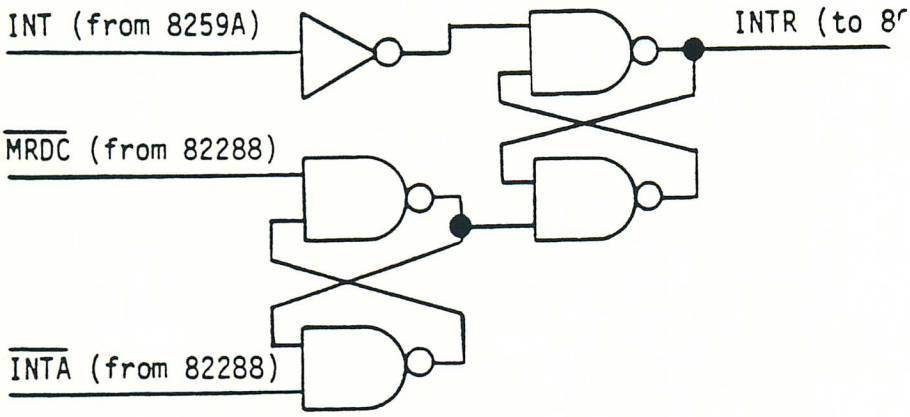
When the INTR is activated and external interrupts are enabled, INTR must be held active until the CPU performs the first INTA bus cycle to process the external interrupt. Failure to keep the INTR active until the first INTA cycle occurs could cause unpredictable CPU behavior. When this errata is corrected, failure to keep the INTR input active will only possibly prevent the external interrupt from being recognized.

Some I/O devices may allow their interrupt output output to go inactive after previously signalling an interrupt. For example, the 8253 or 8254 timer used for real-time clocks can remove the interrupt request if not serviced early enough (even if the 8259A interrupt controller is in edge-triggered mode).

When using the 8259A Priority Interrupt controller, below are several cases in which an active 8259A INT output will go inactive, possibly before the 80286 CPU generates its first INTA cycle to process the interrupt. Both cases below can be handled if the 80286 interrupts are always disabled before programming the interrupt controller.

1) The interrupt request seen by the CPU can also be removed by the 8259A interrupt controller even though the interrupt from the I/O device remains active. For example, system software may mask an 8259A interrupt input just after the I/O device asserts it. The net effect is an INTR signal at the 80286 that goes active then inactive. (even if the 8259A is in edge-triggered mode),
2) the 8259A INT output will go inactive for about the duration of the write pulse when the CPU writes an OCW3 (operation control word 3) which selects the IRR (interrupt request register),

Alternatively, below is a circuit which handles any interrupt removal problems without any software changes. The circuit below latches the 8259A INT output before it is sent to the 80286 INTR input. The latch becomes transparent between the first INTA pulse and the first read of the vector from the interrupt vector table.



The default interrupt function of the 8259A will guarantee a proper vector for IR7, the default interrupt, if at the time INTA is performed no unmasked interrupt is active. The interrupt handler for IR7 can read ISR7 of the 8259A to tell if this was a real interrupt on IR7. The ISR7 bit will be 0 if at the time the interrupt was acknowledged, no unmasked IR inputs of the 8259A were active ("phantom" interrupt).