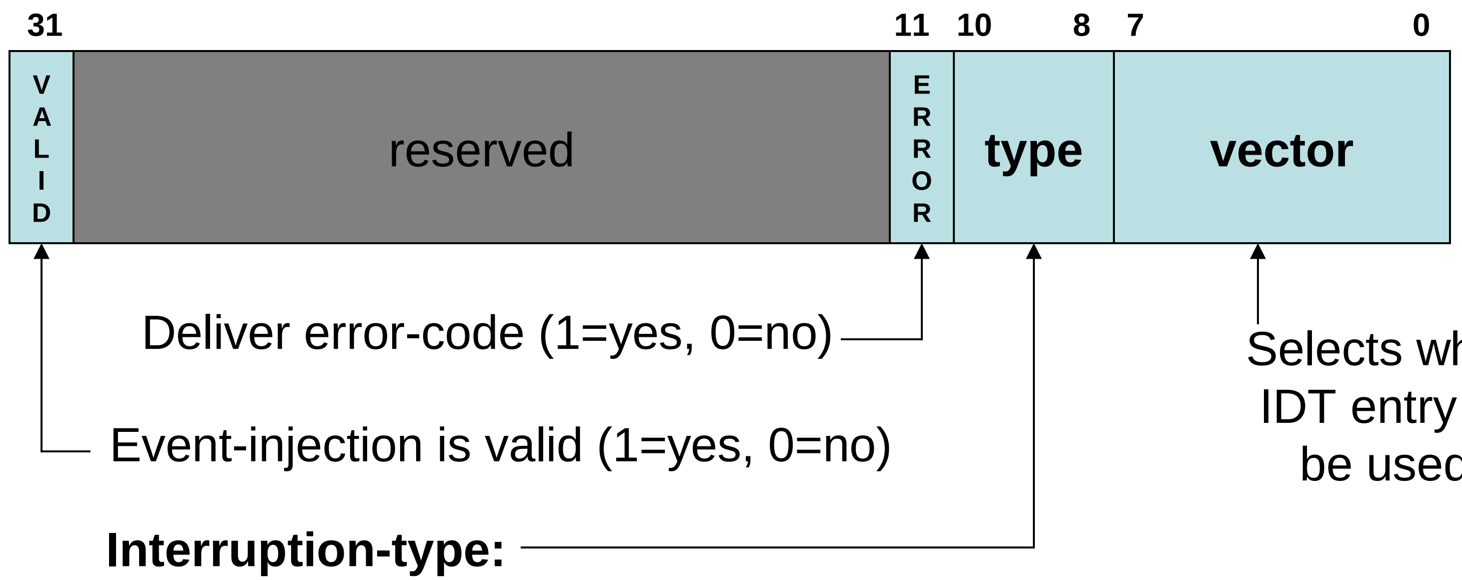
Using VM controls

Examples of ‘event-injection’ by our ‘host’ VMM into its ‘guest’ VM

Document ambiguities

- As is usually the case with documentation for complex computer hardware, manuals intended to explain its operations tend to omit some details
- Here we highlight some examples for the Intel ‘Virtualization Technology’ manuals pertaining to the ‘event-injection’ facility
- Experimentation can help clarify questions

VM-entry interruption

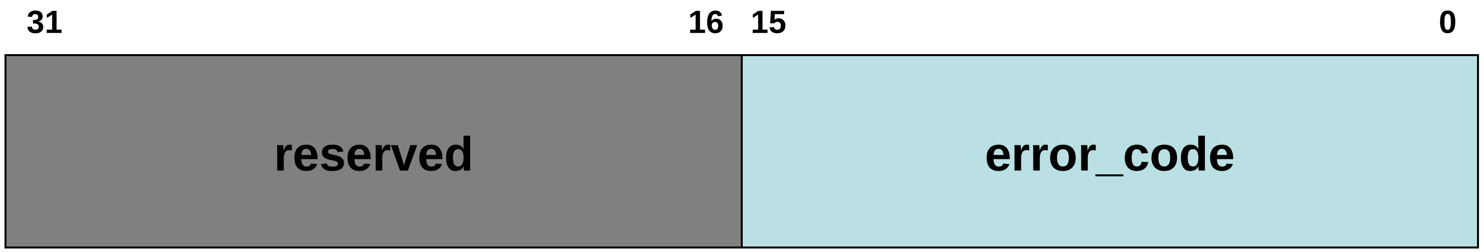

control_VMentry_interruption_information
31
11
10
8
7
0
VALID
reserved
ERROR
type
vector
Deliver error-code (1=yes, 0=no)
Event-injection is valid (1=yes, 0=no)
Selects which IDT entry will be used

Interruption-type:

0 = External Interrupt
1 = (reserved)
2 = Non-Maskable Interrupt
3 = Hardware Exception
4 = Software Interrupt
5 = Privileged software exception
6 = Software exception
7 = (reserved)

Error-code delivery

control_VMentry_interruption_error_code

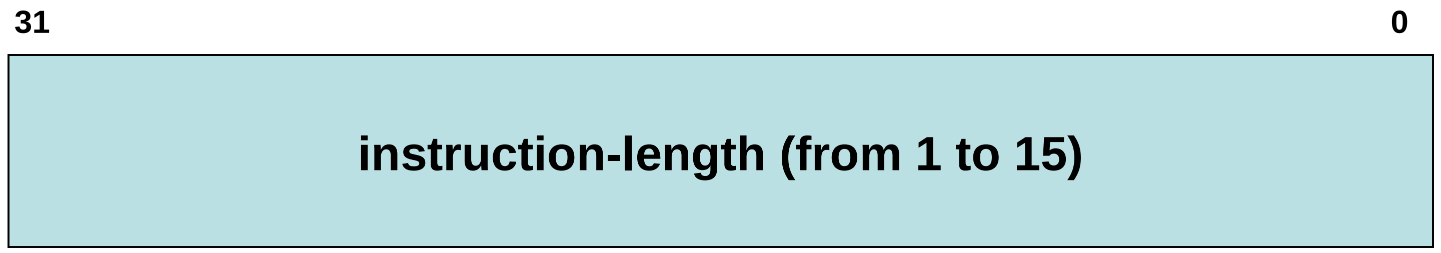


Upon VM entry this error-code will be pushed onto the guest’s stack if and only if, in the control_VMentry_interruption_information field, the VALID bit (bit 31) and the DELIVER ERROR-CODE bit (bit 12) both are set to 1

QUESTION: Are any checks performed on the error_code’s format? Is it indeed required to be just 16-bits (as is suggested here)?

Software events

control_VMentry_instruction_length



For injection of events whose type is 4, 5, or 6, this field is used by the cpu to determine what value for register RIP will be pushed on the guest's stack

Type 4: Software interrupt
Type 5: Privileged software exception
Type 6: Software exception

QUESTION: What would be an example of each of these event types?

Our demo program

- We created an LKM (named ‘inject08.c’) that injects an event of type 0 (External Interrupt) into our ‘guest’ Virtual Machine (if the IF-bit in its EFLAGS register is set)
- The real-mode BIOS interrupt-handler for interrupt-vector 8 is known to increment the Timer-Tick Counter at address 0x46C
- Our ‘seeevent.cpp’ program shows that!

In-class exercise

- Explore the unanswered questions using our ‘inject08.c’ device-driver module, and our ‘seeevent.cpp’ application-program, by introducing modifications into those files that will test your various hypotheses