UNIT II

PART-II

Systems Architecture

Topics To Cover Part II

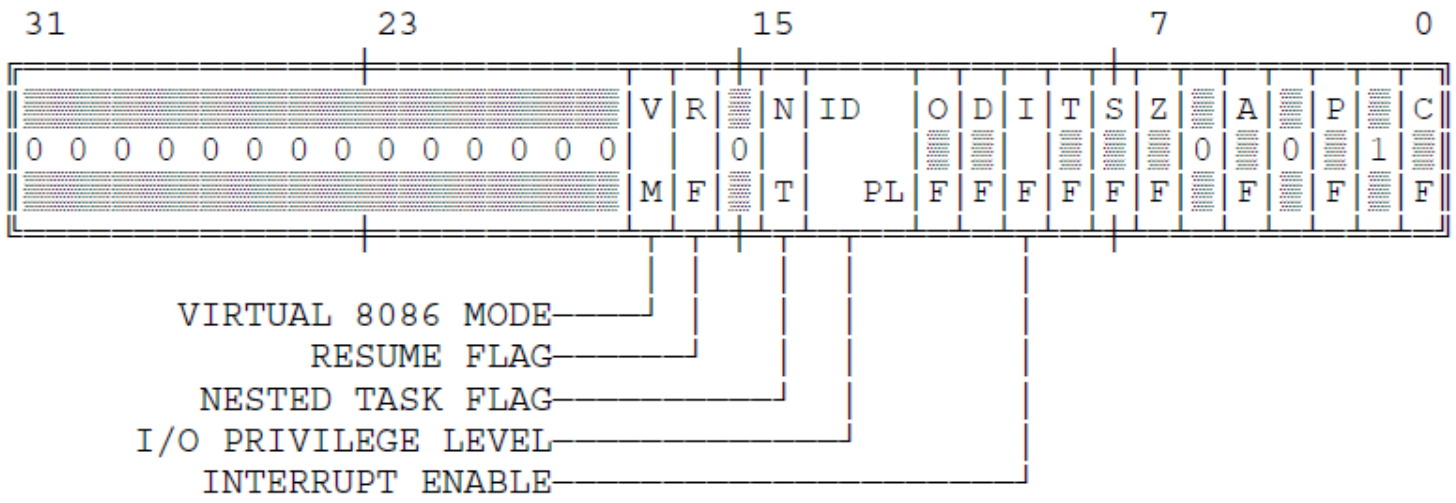
Systems Architecture:

- Systems Registers
- Systems Instructions

Systems Registers

- The registers designed for use by systems programmers fall into these classes:

1.EFLAGS
2.Memory-Management Registers
3.Control Registers
4.Debug Registers
5.Test Registers



NOTE

0 OR 1 INDICATES INTEL RESERVED. DO NOT DEFINE.

System Flags of EFLAGS Register

Systems Flags

The systems flags of the EFLAGS register control

- I/O,
- maskable interrupts,
- debugging,
- task switching,
- and enabling of virtual 8086 execution

in a protected, multitasking environment.

TF (Trap Flag, bit 8)

- Setting TF puts the processor into single-step mode for debugging.
- In this mode, the CPU automatically generates an exception after each instruction, allowing a program to be inspected as it executes each instruction.
- Single-stepping is just one of several debugging features of the 80386.

IF (Interrupt-Enable Flag, bit 9)

- Setting IF allows the CPU to recognize external (maskable) interrupt requests.
- Clearing IF disables these interrupts.
- IF has no effect on either exceptions or nonmaskable external interrupts.

IF (Interrupt-Enable Flag, bit 9)

- Setting IF allows the CPU to recognize external (maskable) interrupt requests.
- Clearing IF disables these interrupts.
- IF has no effect on either exceptions or nonmaskable external interrupts.

IOPL (Input / Output Privilege Level Flags, bits 12-13)

- Used in protected mode to generate 4 levels of security

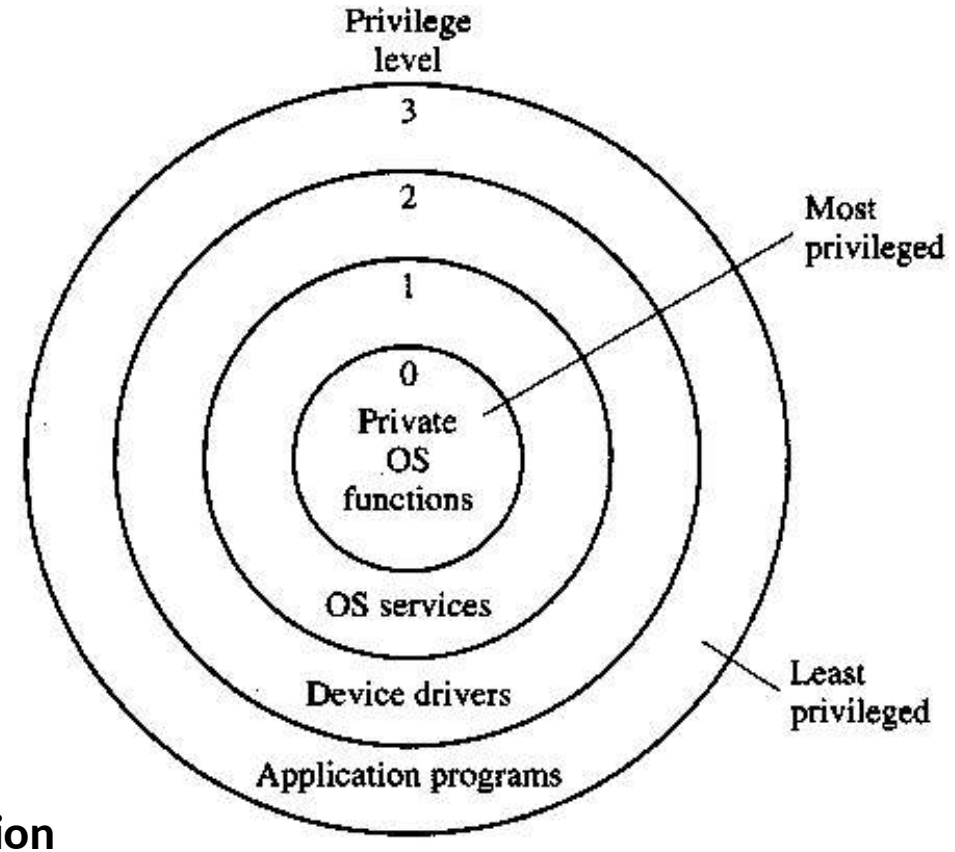


Fig: Rings of Protection

NT (Nested Task, bit 14)

- The processor uses the nested task flag to control chaining of interrupted and called tasks.
- NT influences the operation of the IRET instruction.
- Used in protected mode
- When set, indicates that one system task has invoked another via a CALL instruction, rather than a JMP.

RF (Resume Flag, bit 16)

- The RF flag temporarily disables debug exceptions so that an instruction can be restarted after a debug exception without immediately causing another debug exception.

VM (Virtual 8086 Mode, bit 17)

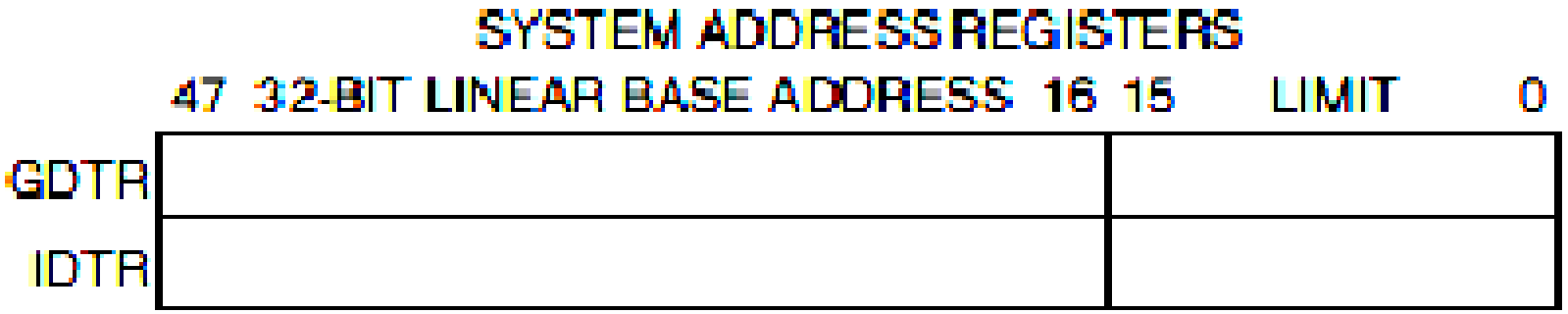
- When set, the VM flag indicates that the task is executing an 8086 program.

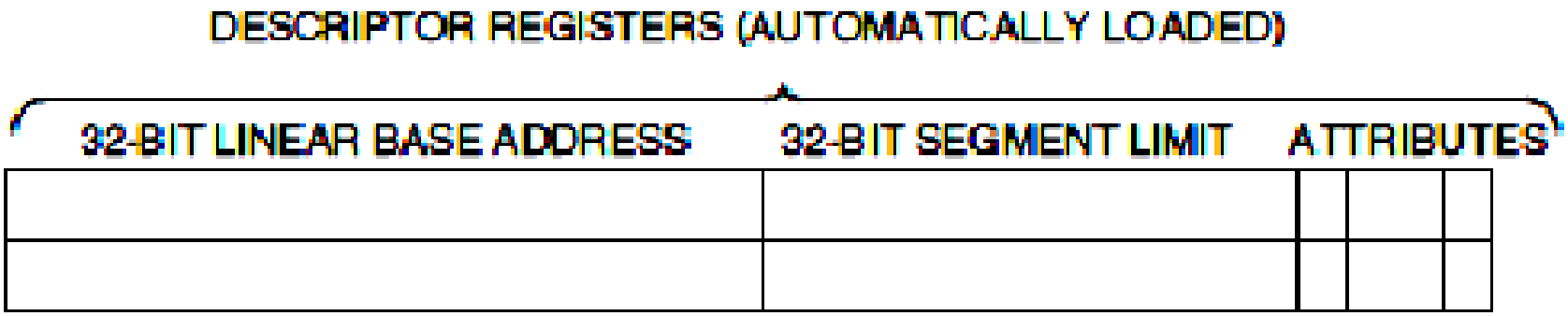
Memory-Management Registers

- Four registers of the 80386 locate the data structures that control segmented memory management:

1. GDTR Global Descriptor Table Register
2. LDTR Local Descriptor Table Register
   - These registers point to the segment descriptor tables GDT and LDT.
3. IDTR Interrupt Descriptor Table Register
   - This register points to a table of entry points for interrupt handlers (the IDT).
4. TR Task Register
   - This register points to the information needed by the processor to define the current task.

GDTR and IDTR

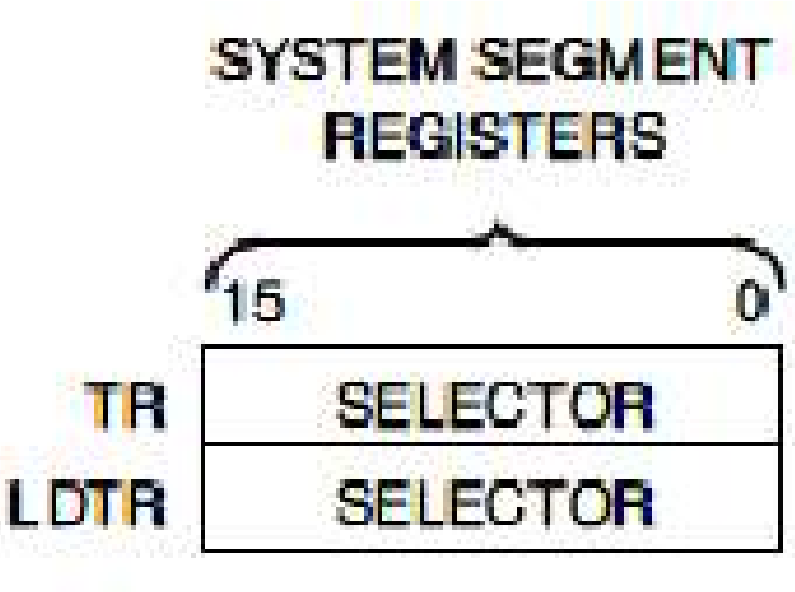




- The remaining bytes of these registers are then loaded automatically by the processor from segment descriptor referenced by the operand.
- Descriptor Registers are not visible to programmer

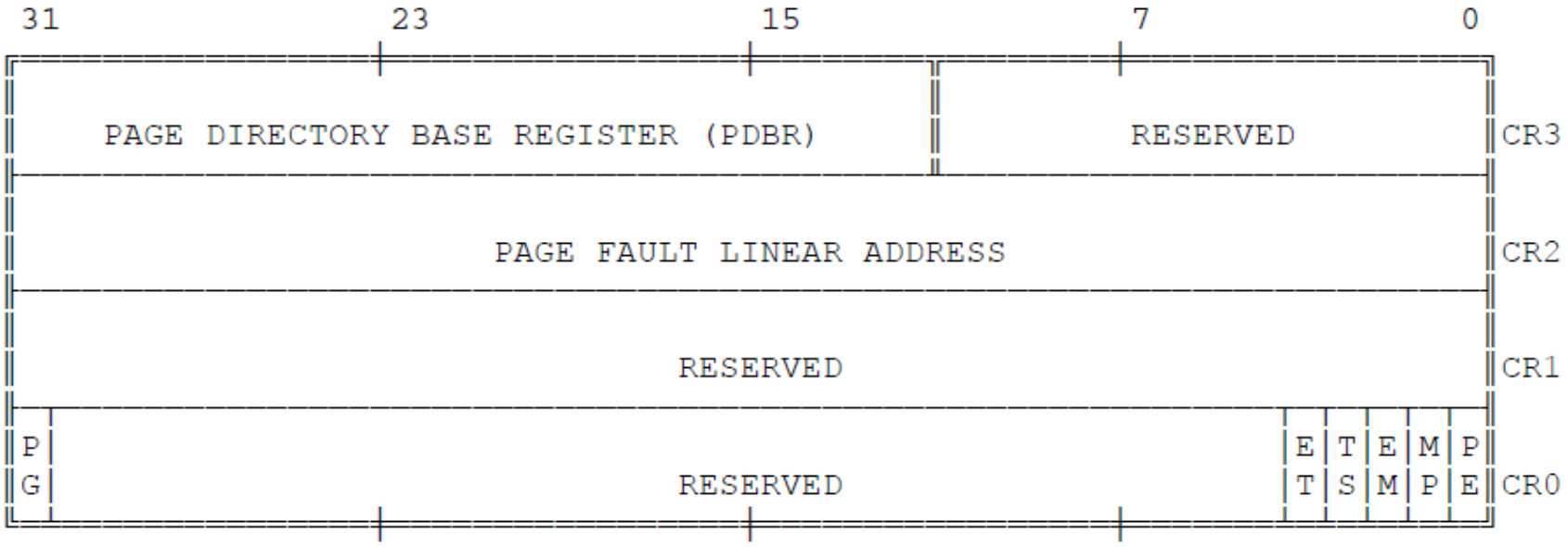
LDTR and TR

- LDTR and TR can be loaded with instructions which takes a 16-bit segment selector as an operand.



Control Registers

- 80386 control registers CR0, CR2, and CR3
- CR1 is reserved
- These registers are accessible to systems programmers only via variants of the MOV instruction, which allow them to be loaded from or stored in general registers



Control Registers

Control Registers

- Intel386 DX has Four control registers of 32 bits- CR0, CR2 and CR3 - to hold machine state of a global nature
- These registers along with System Address Registers hold machine state that affects all tasks in the system
- CR0 contains 6 defined bits for control and status purposes.

CR0 : Machine Control Register

- The low-order 16 bits of CR0 is defined as **Machine Status Word**
- To operate only on the low-order 16-bits of CR0, LMSW and SMSW instructions are used.
- For 32-bit operations the system should use MOV CR0, Reg instruction.

CR0

- contains system control flags, which control or indicate conditions that apply to the system as a whole, not to an individual task.

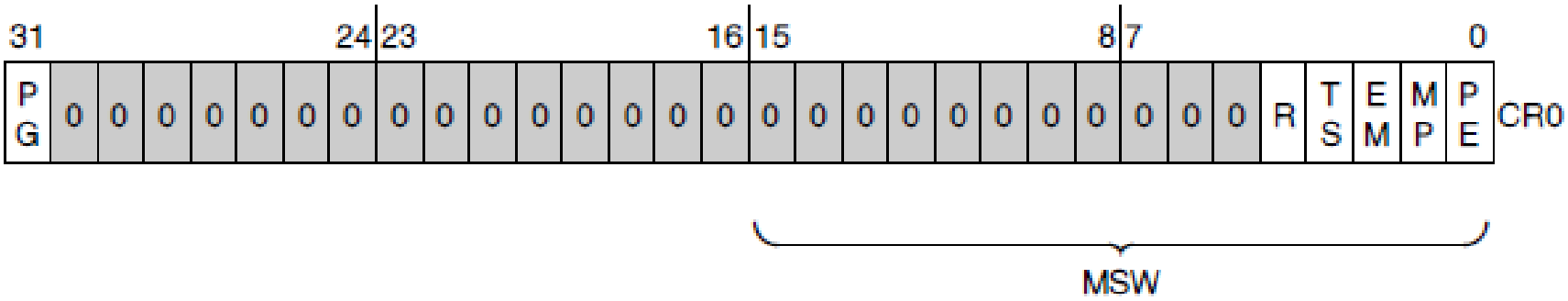
CR0 bits

1. EM (Emulation, bit 2) :EM indicates whether coprocessor functions are to be emulated.
2. ET (Extension Type, bit 4): ET indicates the type of coprocessor present in the system (80287 or 80387).
3. MP (Math Present, bit 1) : MP controls the function of the WAIT instruction, which is used to coordinate a coprocessor.

CR0

4. PE (Protection Enable, bit 0) : Setting PE causes the processor to begin executing in protected mode. Resetting PE returns to real-address mode.
5. PG (Paging, bit 31) : PG indicates whether the processor uses page tables to translate linear addresses into physical addresses.
6. TS (Task Switched, bit 3) : The processor sets TS with every task switch and tests TS when interpreting coprocessor instructions.

CR0 : Machine Control Register



- **Bit 31 (PG Bit, Paging Enable)** : The PG bit is set to enable the on-chip paging unit.
- **Bit 4 (Reserved) (ET bit, Extension Type)** : When power is applied , 80386 detects whether numeric processor connected is 80287 or 80387 and ET to 1, if numeric processor is 80387 this is necessary because 80387 uses a slightly different protocol than 80287

CR0 : Machine Control Register

- **Bit 3 (TS Bit, Task Switched) :**
  - TS is automatically set whenever a task switch operation is performed.
- **Bit 2 (EM Bit, Emulate Coprocessor) :**
  - This bit indicates whether coprocessor functions are to be emulated
  - If EM is set , the processor signals Coprocessor Not Available fault (exception 7 and exception 11 device not available).

CR0 : Machine Control Register

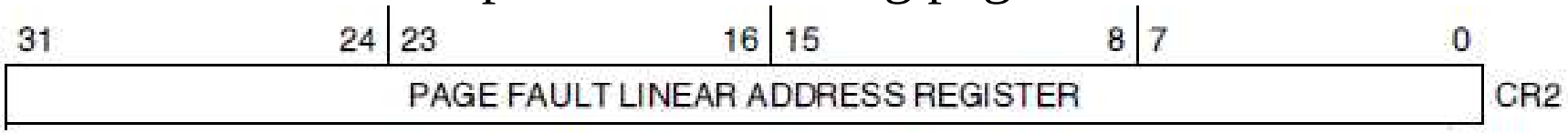
- Bit 1 (MP Bit, Monitor Coprocessor) :
- It indicates whether coprocessor is actually attached.
- If , when executing a WAIT instruction CPU finds MP set, then it tests TS flag.
- When both MP = 1 and TS= 1, the WAIT opcode signals exception 7.
- Bit 0 (PE Bit, Protection Enable) :
- The PE bit is set to enable the Protected Mode.
- If PE is reset, the processor operates in Real Mode.

CR2

- CR2 is used for handling page faults when PG is set.
- The processor stores in CR2 the linear address that triggers the fault.

CR2 : Page Fault Linear Address

- CR2 holds the 32-bit linear address that caused the last page fault detected.
- When page fault occurs the 80386 generates an exception 14 (page fault)
- This address is important for writing page fault routine .

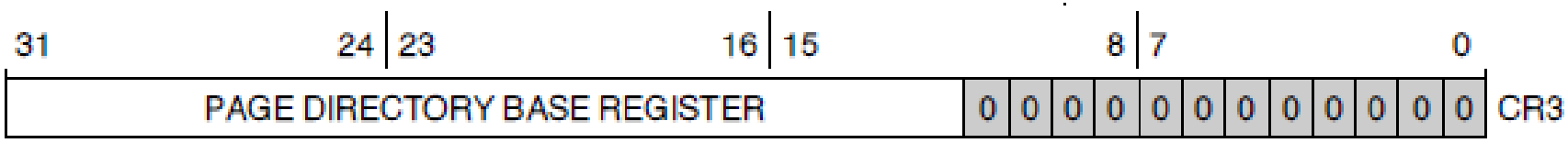


CR3

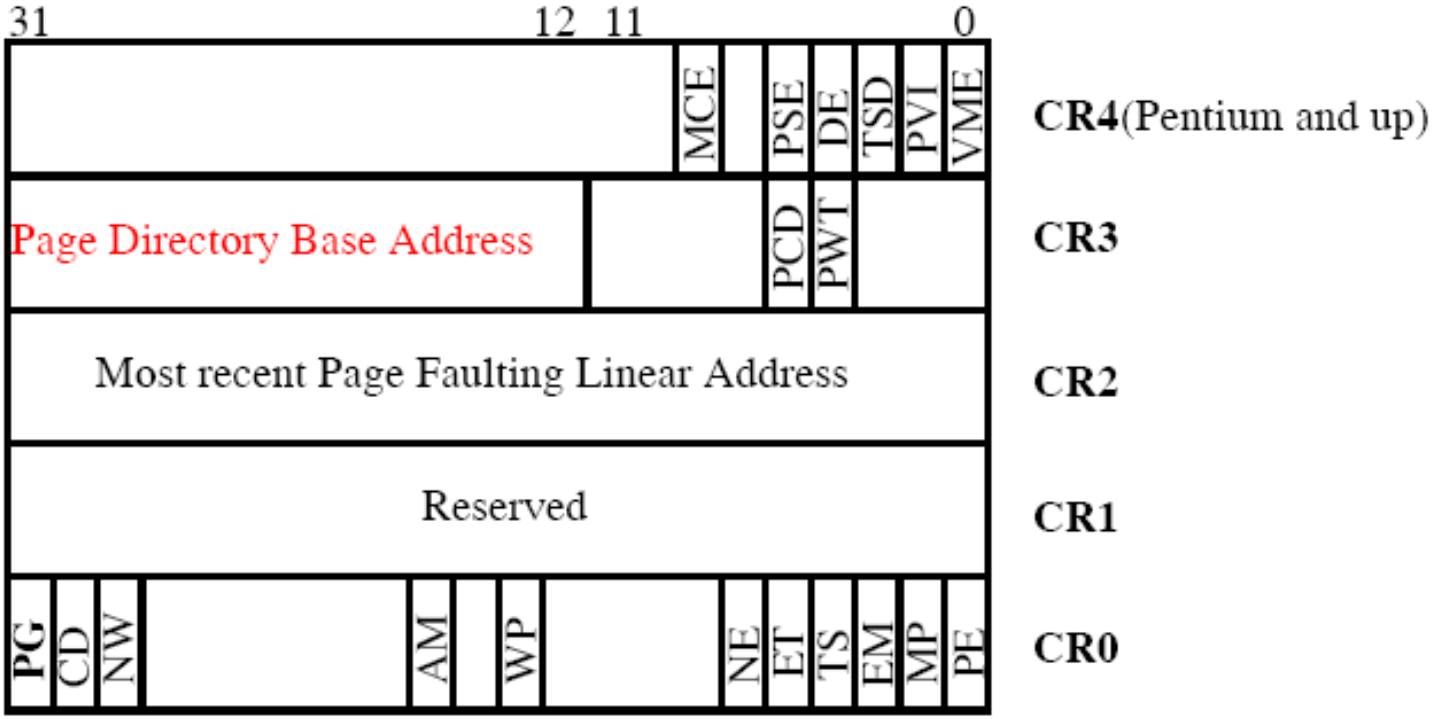
- CR3 is used when PG is set.
- CR3 enables the processor to locate the page table directory for the current task.

CR3 : Page Directory Base Address

- CR3 contains the physical base address of the page directory table.
- The Intel386 DX page directory table is always page-aligned (4 Kbyte-aligned).
- Thus the lowest twelve bits of CR3 are ignored.
- A task switch through a TSS invalidates all page table entries in paging unit cache.



The **paging unit** is controlled by the microprocessors control registers:



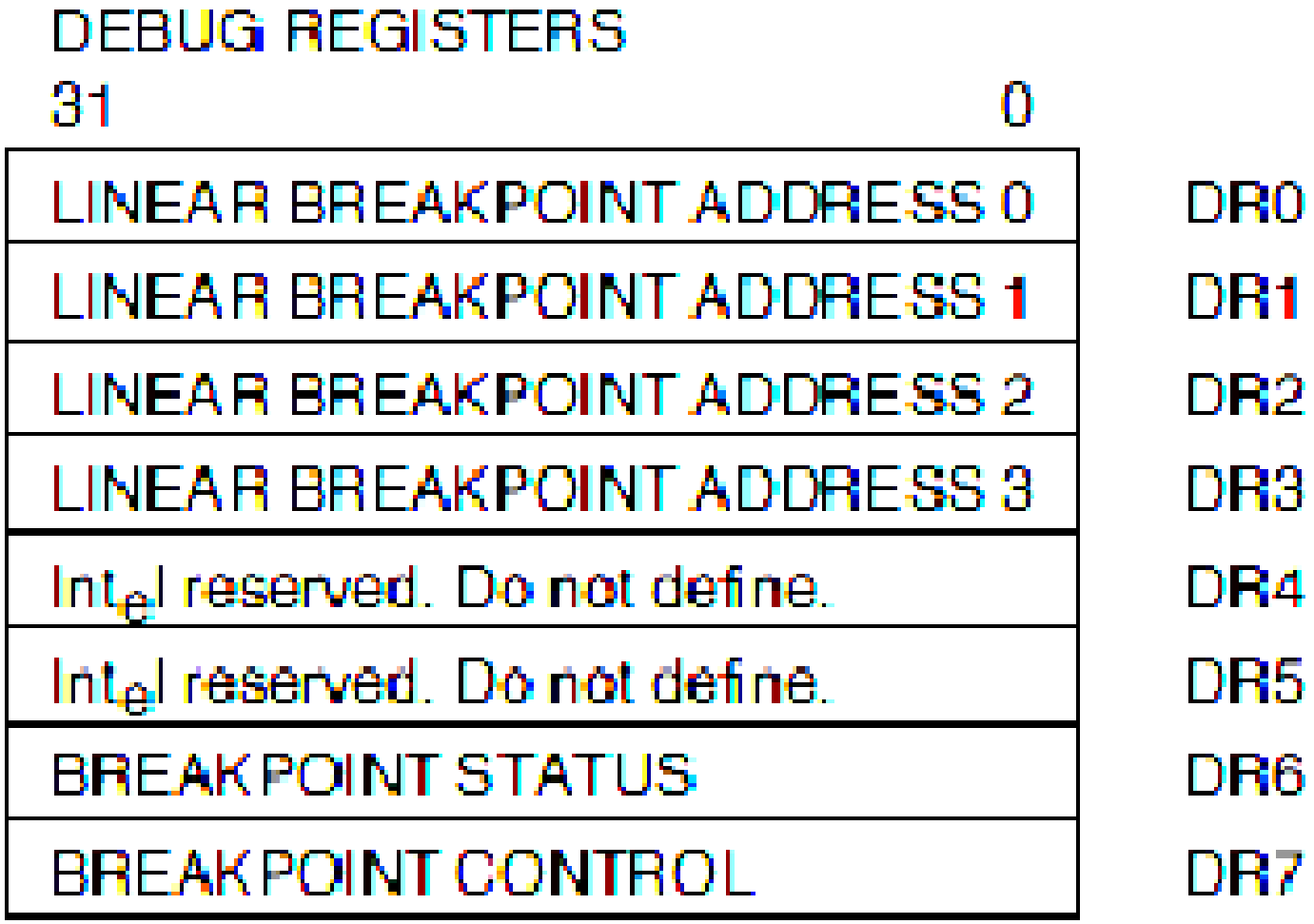
Debug Register

- The debug registers bring advanced debugging abilities to the 80386, including data breakpoints and the ability to set instruction breakpoints without modifying code segments.

Debug Registers

- Debugging of 80386 allows data access breakpoints as well as code execution breakpoints.
- 80386 contains total 8 reg but 6 debug registers to specify
  - 4 breakpoints
  - Breakpoint Control options
  - Breakpoint Status

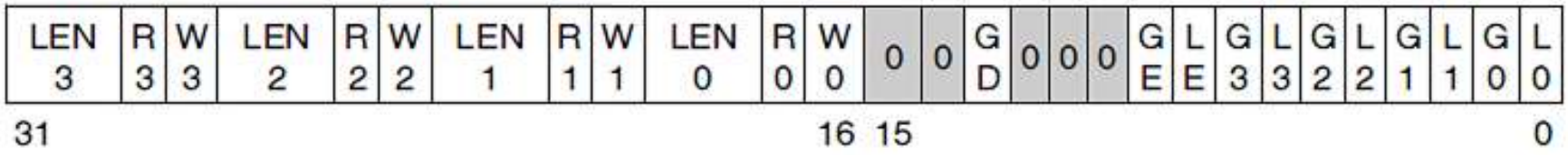
Debug Registers



Linear Breakpoint Address Registers

- The breakpoint addresses specified are 32-bit linear addresses
- While debugging, Intel 386 h/w continuously compares the linear breakpoint addresses in DR0-DR3 with the linear addresses generated by executing software.

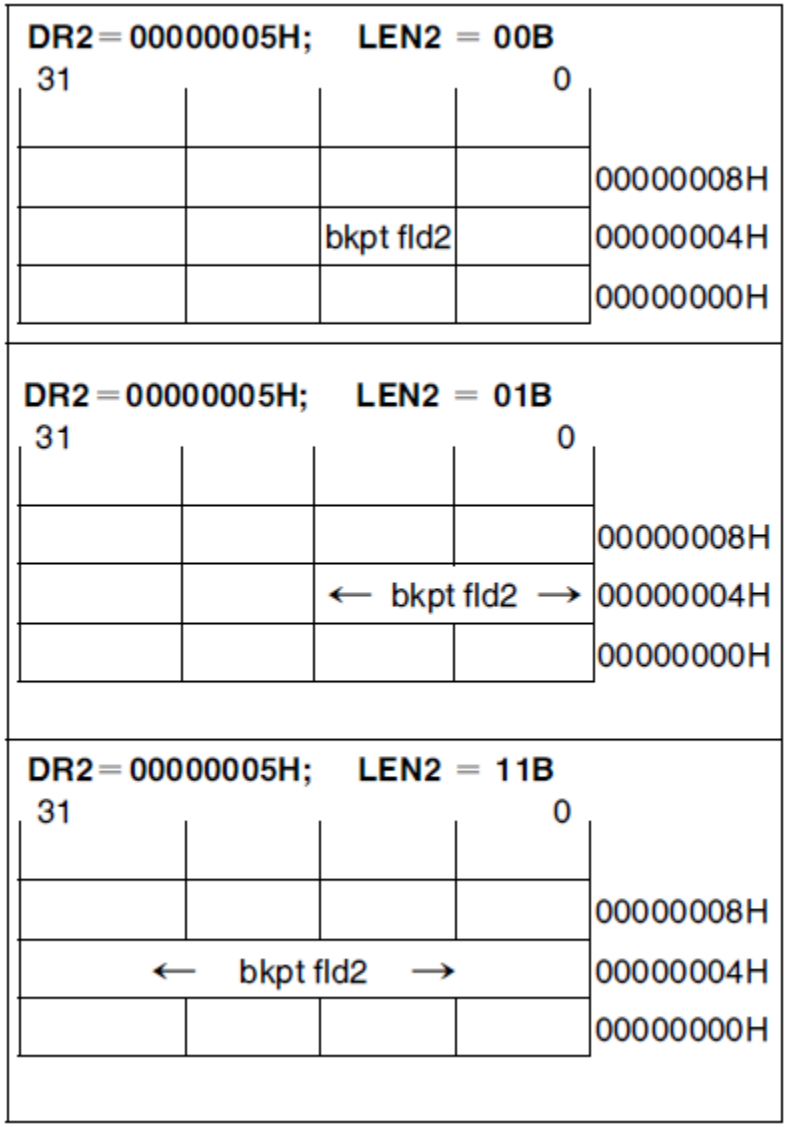
Debug Control Register



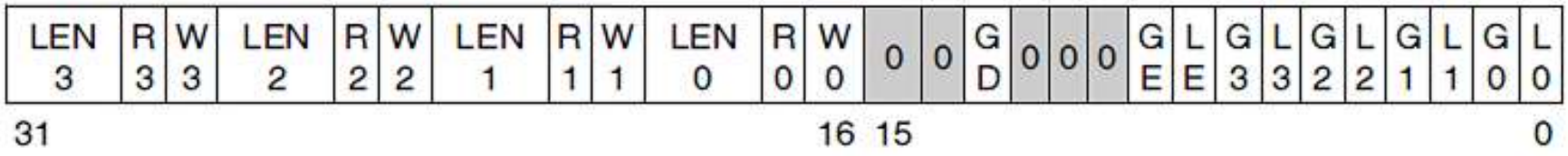
- LENi(i=0 - 3): Breakpoint Length Specification Bits:
  - 2 bit field for each breakpoint
  - Specifies length of breakpoint fields
  - The choices of data breakpoints are 1byte, 2bytes & 4bytes
  - For instruction execution breakpoint, the length is 1(beginning byte address)

LEN_i Encoding

LENi Encoding	Breakpoint Field Width	Usage of Least Significant Bits in Breakpoint Address Register i, (i = 0 – 3)
00	1 byte	All 32-bits used to specify a single-byte breakpoint field.
01	2 bytes	A1–A31 used to specify a two-byte, word-aligned breakpoint field. A0 in Breakpoint Address Register is not used.
10	Undefined—do not use this encoding	
11	4 bytes	A2–A31 used to specify a four-byte, dword-aligned breakpoint field. A0 and A1 in Breakpoint Address Register are not used.



Debug Control Register

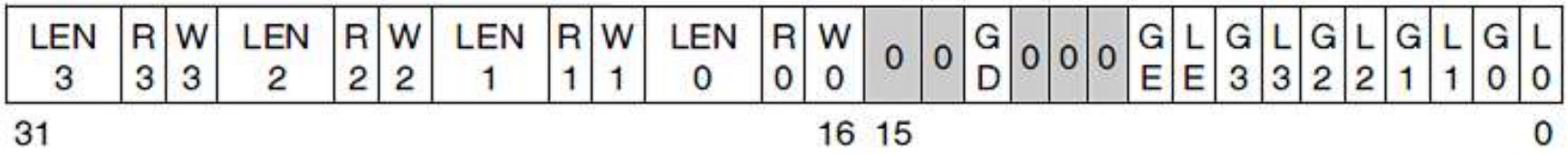


- RWi(i=0 - 3): Memory Access Qualifier Bit
  - 2 bit field for each breakpoint
  - Specifies the type of usage which must occur in order to activate the associated breakpoint

Debug Registers

RW Encoding	Usage Causing Breakpoint
00	Instruction execution only
01	Data writes only
10	Undefined—do not use this encoding
11	Data reads and writes only

Debug Control Register



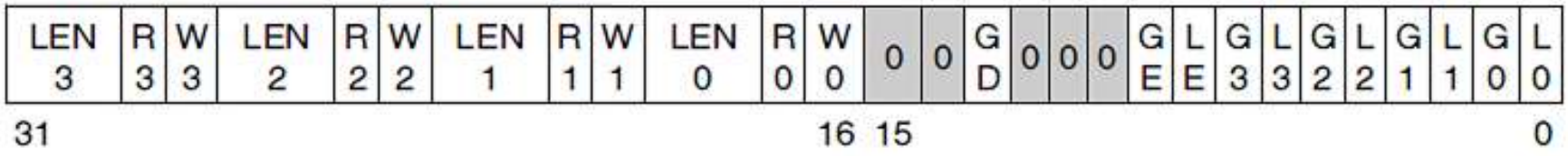
- GD: Global Debug Register Access Detect
  - Debug registers can only be accessed in real mode or at privilege level 0 in protected mode
  - GD bit, when set, provides extra protection against any Debug Register access even in Real Mode or at privilege level 0 in Protected Mode.

Debug Control Register

GD: Global Debug Register Access Detect

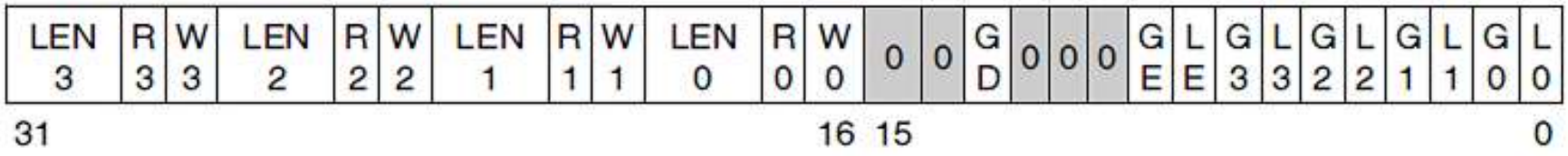
- This additional protection feature is provided to guarantee that a software debugger can have full control over the Debug Register resources when required.
- The GD bit, when set, causes an exception 1 fault if an instruction attempts to read or write any Debug Register.
- The GD bit is then automatically cleared when the exception 1 handler is invoked, allowing the exception 1 handler free access to the debug registers.

Debug Control Register



- **GE and LE bit:** Exact data breakpoint match, global and local
- If either GE or LE is set, any data breakpoint trap will be reported exactly after completion of the instruction that caused the operand transfer.
- LE bit is **cleared during task** switch and is used for task-local breakpoints.
- GE bit is **unaffected during a task switch** and remain enabled during all tasks executing in the system.

Debug Control Register



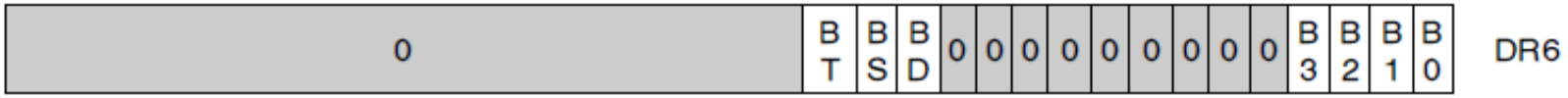
- Gi and Li(i=0 - 3): Breakpoint Enable, global and local
  - If either Gi and Li is set then the associated breakpoint is enabled.

Debug Status Register

- A Debug Status Register allows the exception 1 handler to easily determine why it was invoked.
- It can be invoked as a result of one of several events:

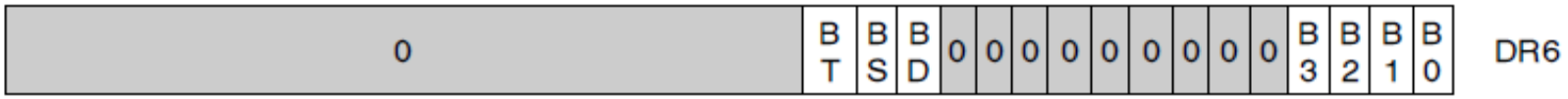
1) DR0 Breakpoint fault/trap.
2) DR1 Breakpoint fault/trap.
3) DR2 Breakpoint fault/trap.
4) DR3 Breakpoint fault/trap.
5) Single-step (TF) trap.
6) Task switch trap.
7) Fault due to attempted debug register access when GD = 1.

Debug Status Register



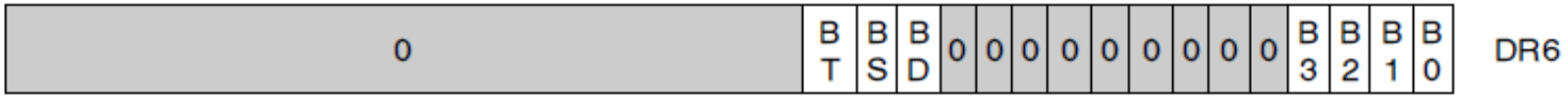
- **B_i :** Debug fault/trap due to breakpoint 0 -3
- Four breakpoint indicator flags, B0-B3, correspond one-to-one with the breakpoint registers in DR0-DR3.
- A flag Bi is set when the condition described by DRi, LENi, and RWi occurs.

Debug Status Register



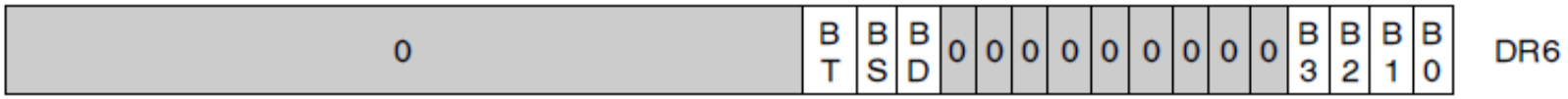
- **BD** : Debug fault due to attempted register access when GD bit is set
  - This bit is set if the exception 1 handler was invoked due to an instruction attempting to read or write to the debug registers when GD bit was set.

Debug Status Register



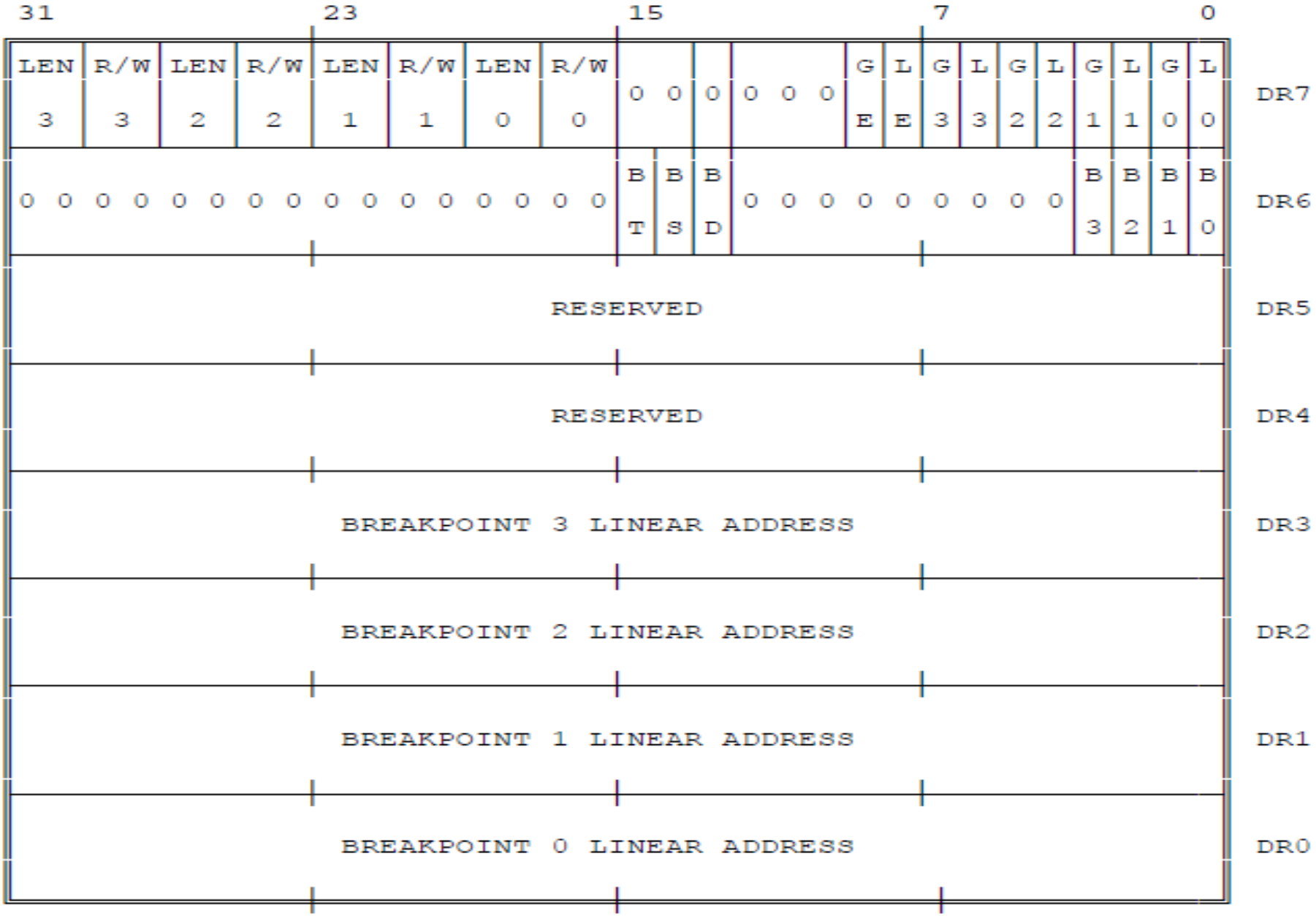
- **BS** : Debug trap due to single step
  - This bit is set if the exception 1 handler was invoked due to the TF bit in the flag register being set

Debug Status Register



- **BT** : Debug trap due to task switch
  - This bit is set if the exception 1 handler was invoked due to a task switch occurring to a task having an Intel386 DX TSS with the T bit set.

Additional Ends



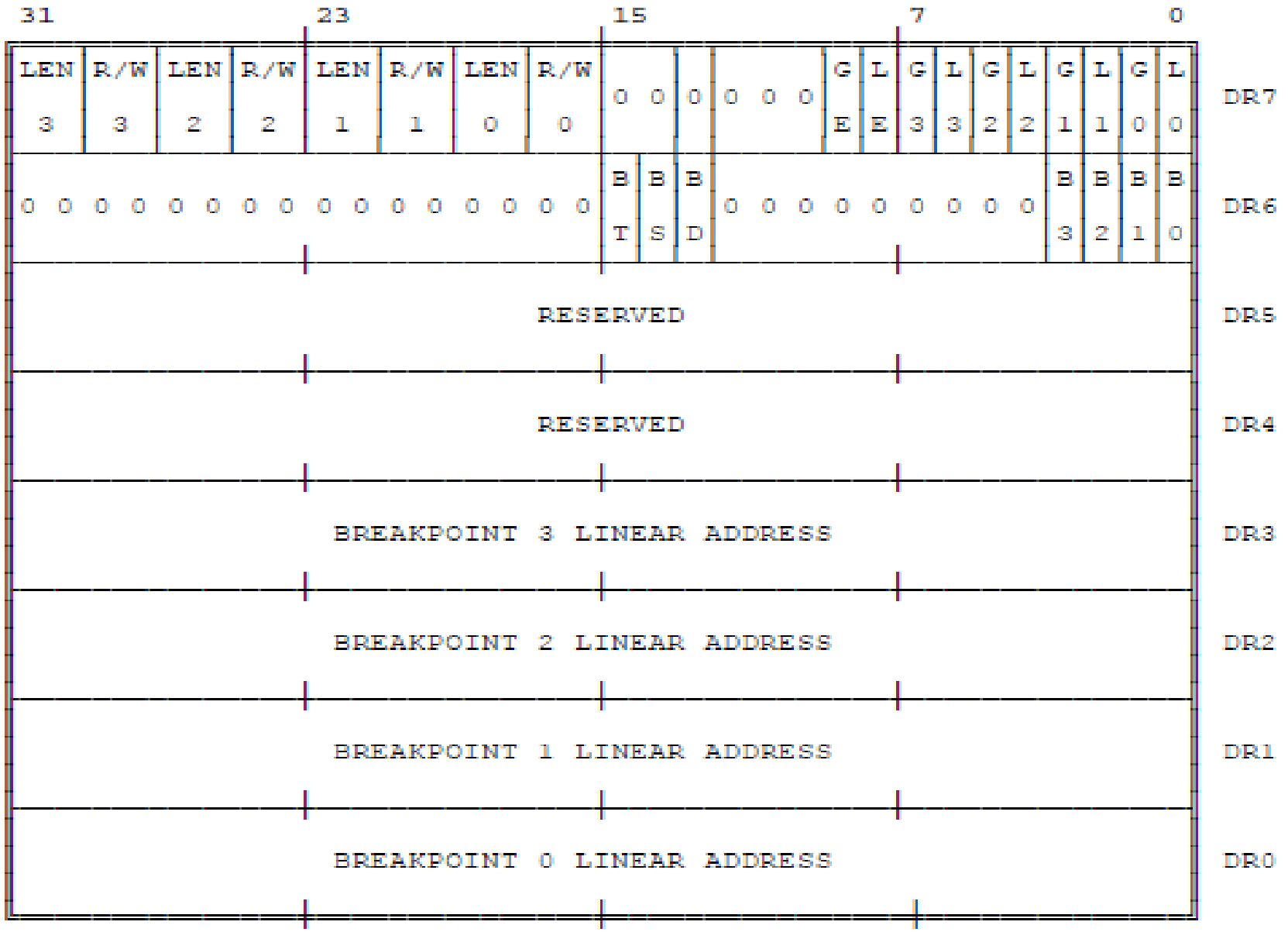
Debug Registers

Debug Address Registers (DR0-DR3)

- Each of these registers contains the linear address associated with one of four breakpoint conditions.
- Each breakpoint condition is further defined by bits in DR7.
- The debug address registers are effective whether or not paging is enabled.
- The addresses in these registers are linear addresses. If paging is enabled, the linear addresses are translated into physical addresses by the processor's paging mechanism.
- If paging is not enabled, these linear addresses are the same as physical addresses.

- when paging is enabled, different tasks may have different linear-to-physical address mappings.
- When this is the case, an address in a debug address register may be relevant to one task but not to another.
- For this reason the 80386 has both global and local enable bits in DR7.
- These bits indicate whether a given debug address has a global (all tasks) or local (current task only) relevance.

Figure 12-1. Debug Registers



NOTE

0 MEANS INTEL RESERVED. DO NOT DEFINE.

Debug Control Register (DR7)

- The debug control register helps

1.to define the debug conditions and

2.selectively enables and disables those conditions.

- For each address in registers DR0-DR3, the corresponding fields R/W0 through R/W3 specify the type of action that should cause a breakpoint.
- The processor interprets these bits as follows:

00 —— Break on instruction execution only

01 —— Break on data writes only

10 —— undefined

11 —— Break on data reads or writes but not instruction fetches

L0 - L3 & G0 – G3 (Local/Global)

- The low-order eight bits of DR7 (L0 through L3 and G0 through G3) selectively enable the four address breakpoint conditions.
- There are two levels of enabling: the local (L0 through L3) and global (G0 through G3) levels.
- The local enable bits are automatically reset by the processor at every task switch to avoid unwanted breakpoint conditions in the new task.
- The global enable bits are not reset by a task switch; therefore, they can be used for conditions that are global to all tasks.

LEN0 – LEN3 (Length of Data)

- 2 bit field each
- Fields LEN0 through LEN3 specify the length of data item to be monitored.
- A length of 1, 2, or 4 bytes may be specified.
- The values of the length fields are interpreted as follows:

00 —— one-byte length
01 —— two-byte length
10 —— undefined
11 —— four-byte length

Note

- If RWn is 00 (instruction execution), then LENn should also be 00.
- Any other length is undefined.

LE & GE (Local Enable / Global Enable)

- The LE and GE bits control the "exact data breakpoint match" feature of the processor.
- If either LE or GE is set, the processor slows execution so that data breakpoints are reported on the instruction that causes them.
- It is recommended that one of these bits be set whenever data breakpoints are armed.
- The processor clears LE at a task switch but does not clear GE.

Debug Status Register (DR6)

- The debug status register permits the debugger to determine which debug conditions have occurred.
- When the processor detects an enabled debug exception, it sets the low-order bits of this register (B0 thru B3) before entering the debug exception handler.
- Bn is set if the condition described by DRn, LENn, and R/Wn occurs.

BT bit

- The BT bit is associated with the T-bit (debug trap bit) of the TSS.
- The processor sets the BT bit before entering the debug handler if a task switch has occurred and the T-bit of the new TSS is set.
- There is no corresponding bit in DR7 that enables and disables this trap; the T-bit of the TSS is the sole enabling bit.

BS bit

- The BS bit is associated with the TF (trap flag) bit of the EFLAGS register.
- The BS bit is set if the debug handler is entered due to the occurrence of a single-step exception.
- The single-step trap is the highest-priority debug exception; therefore, when BS is set, any of the other debug status bits may also be set.

BD bit

- The BD bit is set if the next instruction will read or write one of the eight debug registers and ICE-386 is also using the debug registers at the same time.

1.1.5 **Test Registers**

- The test registers are not a standard part of the 80386 architecture.
- They are provided solely to enable confidence testing of the translation lookaside buffer (TLB), the cache used for storing information from page tables.

1.2 **Systems Instructions**

1. Verification of pointer parameters :

- ARPL — Adjust RPL
- LAR — Load Access Rights
- LSL — Load Segment Limit
- VERR — Verify for Reading
- VERW — Verify for Writing

2. Addressing descriptor tables

- LLDT — Load LDT Register
- SLDT — Store LDT Register
- LGDT — Load GDT Register
- SGDT — Store GDT Register

3. Multitasking :

- LTR — Load Task Register
- STR — Store Task Register

4. Coprocessing and Multiprocessing :

- CLTS — Clear Task-Switched Flag
- ESC — Escape instructions
- WAIT — Wait until Coprocessor not Busy
- LOCK — Assert Bus-Lock Signal

5. Input and Output:

- IN — Input
- OUT — Output
- INS — Input String
- OUTS — Output String

6. Interrupt control:

- CLI — Clear Interrupt-Enable Flag
- STI — Set Interrupt-Enable Flag
- LIDT — Load IDT Register
- SIDT — Store IDT Register

7. Debugging :

- MOV — Move to and from debug registers

8. TLB testing :

- MOV — Move to and from test registers

9. System Control:

- SMSW — Set MSW
- LMSW — Load MSW
- HLT — Halt Processor
- MOV — Move to and from control registers