The 80386 Microprocessor Family

Introduction

- The 80386 family of microprocessors of Intel Corporation is the first 32 bit version of the 8086 family-a switch from 16 bit to 32 bit
- 80386 has upward compatibility with 8086, 8088, 80286 etc
- The 80386 was launched in October 1985, but full-function chips were first delivered in the third quarter of 1986
- Memory management section of 80386 supports the **virtual memory**, **paging** and **four levels of protection**.

rithesh.kini@thadomal.org

Versions of 80386

80386DX – the full version

The first member in 80386 family

this CPU could work with 16-bit and 32-bit external buses.

Comprises of both 32-bit internal registers and 32-bit external bus.

80386SX –the reduced bus version

low cost version of the 80386.

This processor had 16 bit external data bus and 24-bit external address bus.

rithesh.kini@thadomal.org

Features of 80386DX

- The **80386DX** is a 32-bit processor that supports 8-bit/16-bit/32-bit data operands.
- The instruction set is upward compatible with all its predecessors (e.g., 8086).
- With its 32-bit address bus, it can address up to 4Gbytes of physical memory. The physical memory is organized in terms of segments of 4 Gbytes size at maximum.
- The 80386 CPU supports 16K (16384) number of segments and thus the total virtual memory space is 4Gbytes×16K=64 terrabytes.
- The concept of paging is introduced in 80386 that enables it to organize the available physical memory into pages of size 4Kbytes each, under the segmented memory.

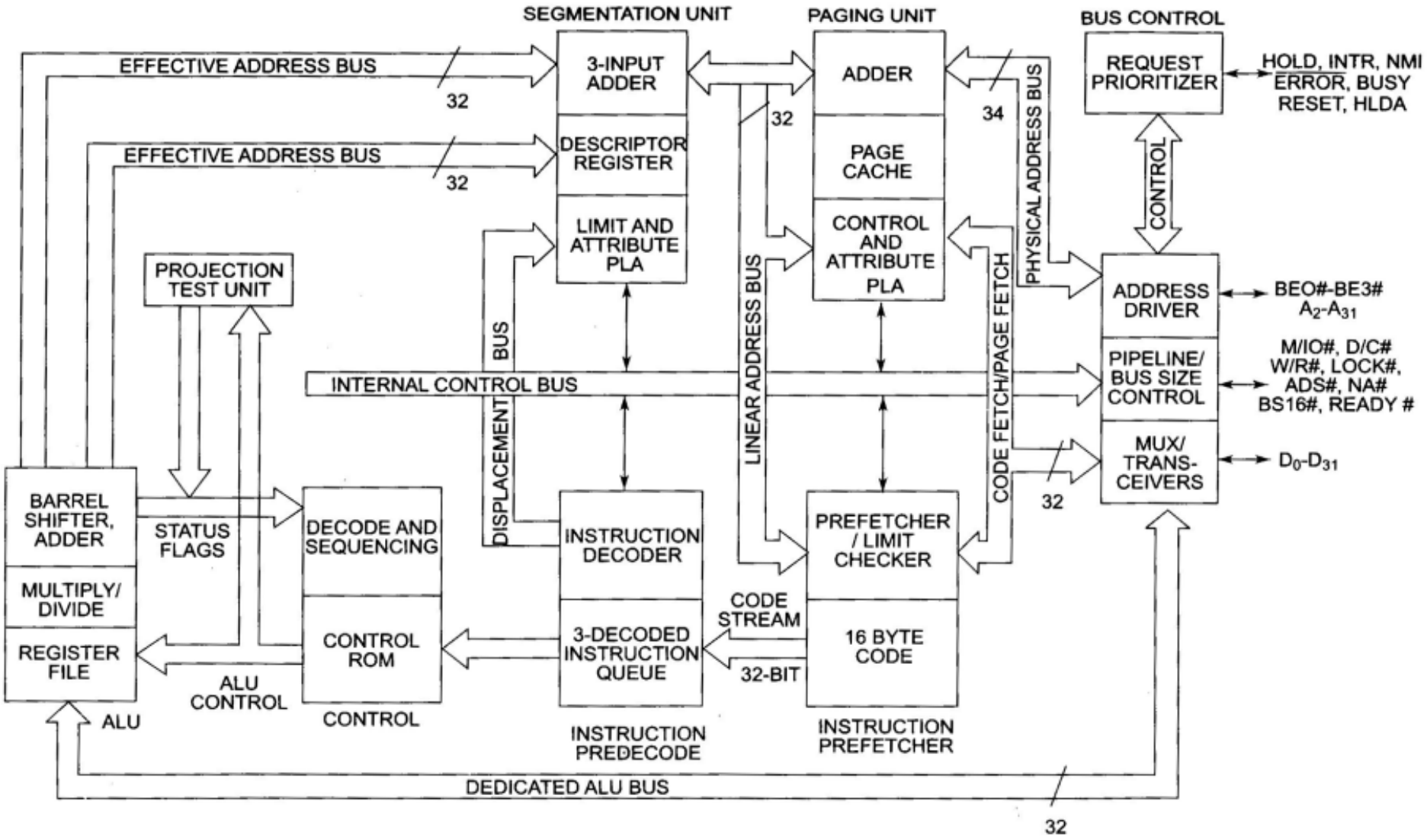
rithesh.kini@thadomal.org

Features of 80386DX

- The 80386 can be **supported by 80387** for mathematical data processing.
- The 80386DX supports **8 debug registers**, for hardware debugging and control.
- The 80386DX has an **on chip address translation cache**.
- Another version **80386SX** has identical architecture but 16-bit data and 24-bit address bus.
- The **80386DX** is available in 132-pin grid array package and has **20MHz** and **33MHz** versions.

rithesh.kini@thadomal.org

Architecture of 80386 microprocessor



80386 Architecture

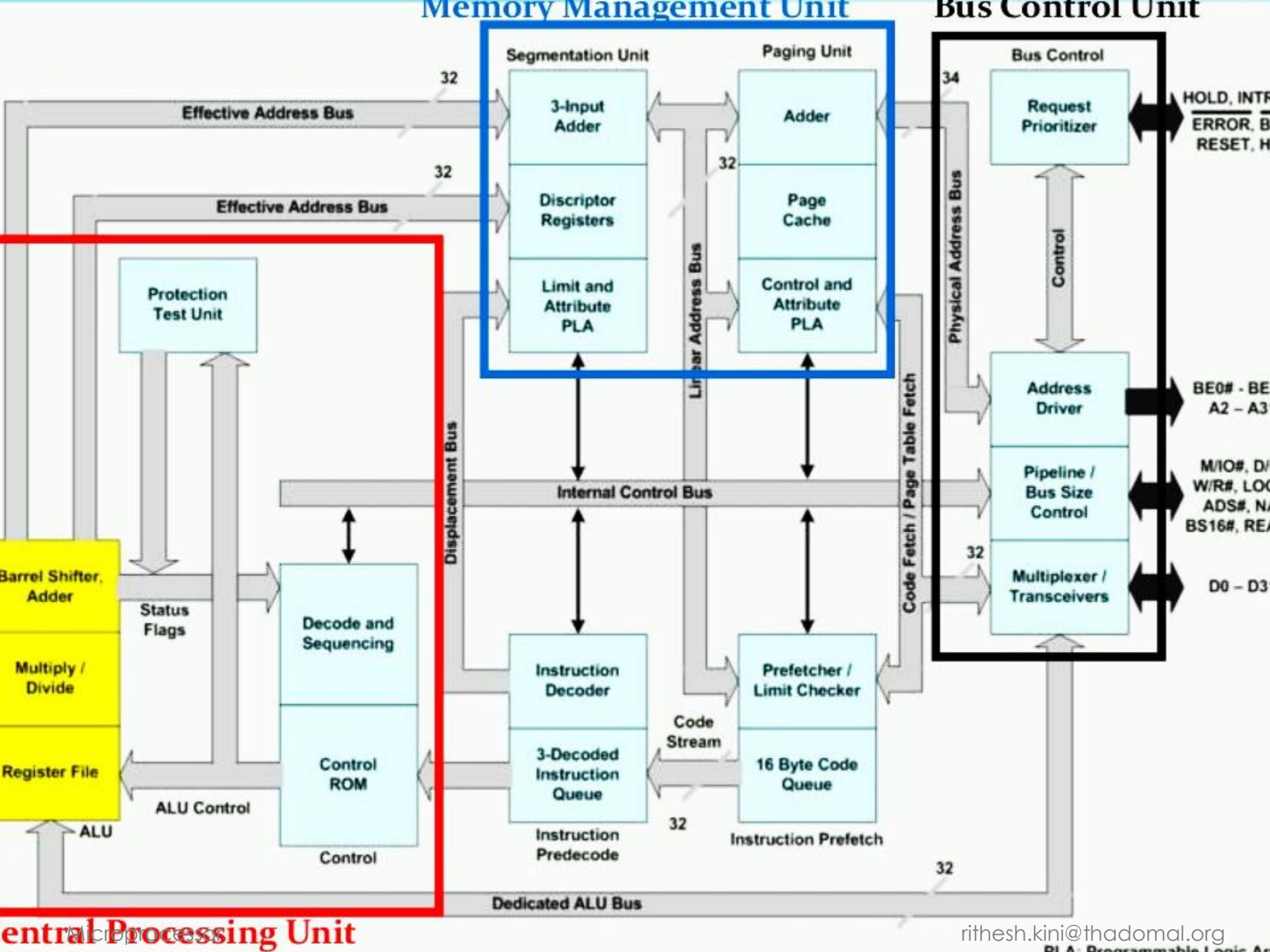
rithesh.kini@thadomal.org

Architecture of 80386 microprocessor

The Internal Architecture of 80386 is divided into 3 sections.

- Central processing unit
  - Prefetcher and prefetch queue
  - Instruction decoder
  - Control ROM and sequencing
  - Execution unit
  - Protection unit
- Bus interface unit
- Memory management unit
  - Segmentation unit
  - Paging unit

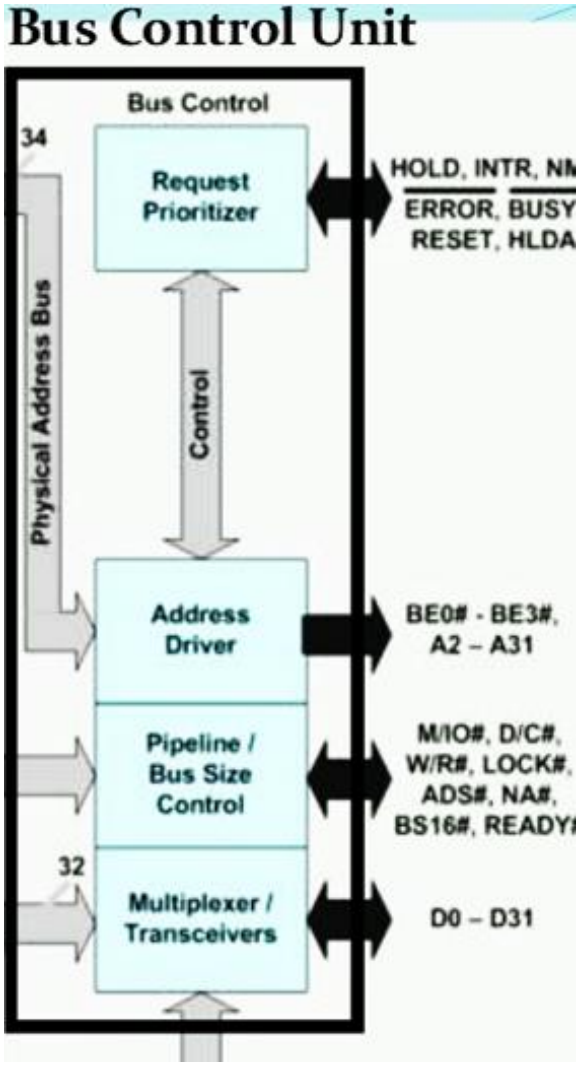
rithesh.kini@thadomal.org



rithesh.kini@thadomal.org

Bus Interface Unit

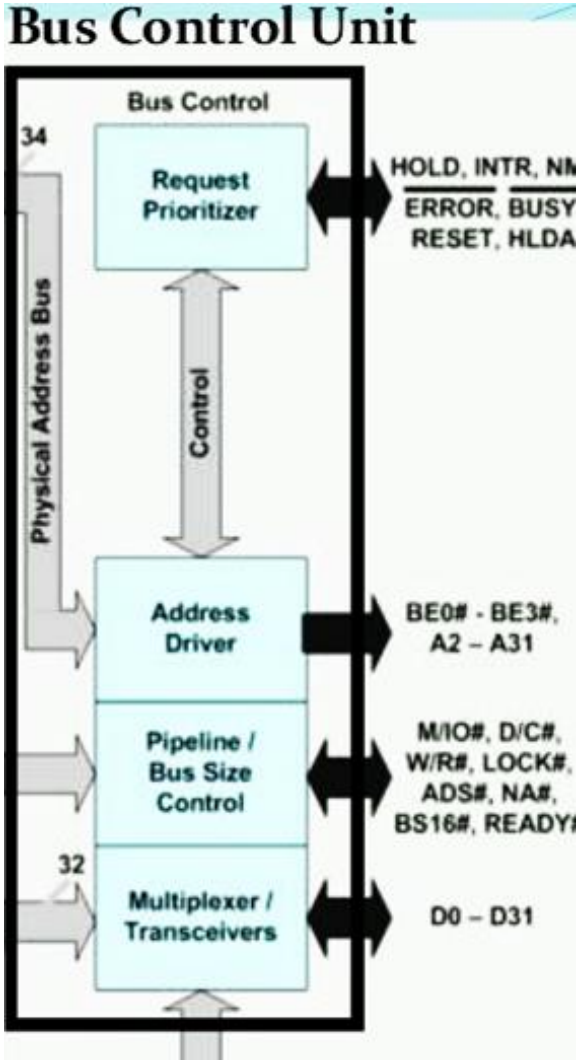
- It is the **interface** to the external devices.
- It provides a full 32-bit bi-directional data bus, a 32-bit address bus and the signals needed to control transfers over the bus.
- In fact, 8-bit, 16-bit and 32-bit data transfers are supported.
- This processing unit contains:
  - latches and drivers for the address bus
  - transceivers for the data bus
  - control logic for signaling whether a memory input/output, or interrupt-acknowledgement bus cycle is to be performed.



rithesh.kini@thadomal.org

Bus Interface Unit

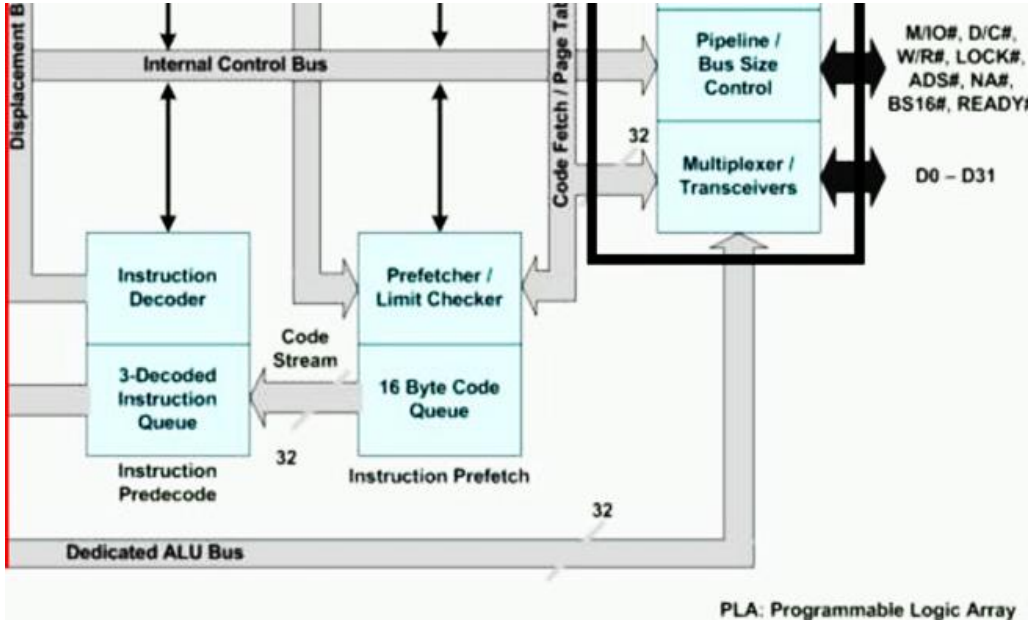
- It accepts internal requests from Prefetch unit for fetching instructions and from the Execution unit for transferring data.
- Prioritizes the internal requests and generates signals to perform bus cycles
- Sends address, data and control signals to communicate with memory and I/O devices.
- Controls the interface to external bus masters and co-processors.



rithesh.kini@thadomal.org

Instruction Unit

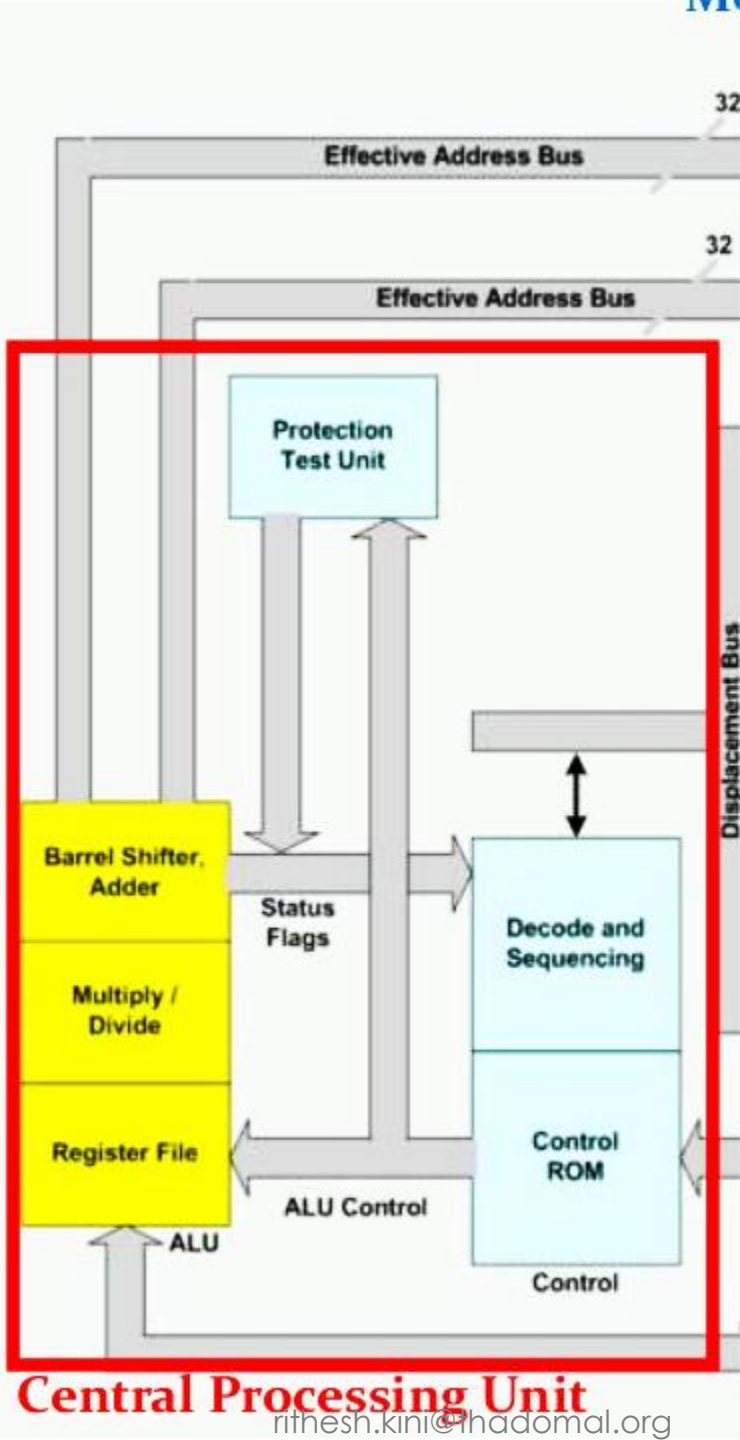
- The **Instruction unit** decodes the opcode bytes received from the 16-byte instruction **code queue** and arranges them in a 3- instruction decoded **instruction queue**.
- After decoding them passes it to the control section for deriving the necessary control signals.



rithesh.kini@thadomal.org

Execution Unit

- The **control ROM** contains the microcode sequences that define the operation performed by each of the machine code instructions.
- The **execution unit** consists of three sub units:
- **1. Control unit:**
  - Contains microcode and special hardware which allows 80386DX to reduce time required for execution of multiply and divide instructions.
  - Speeds the effective address calculation.

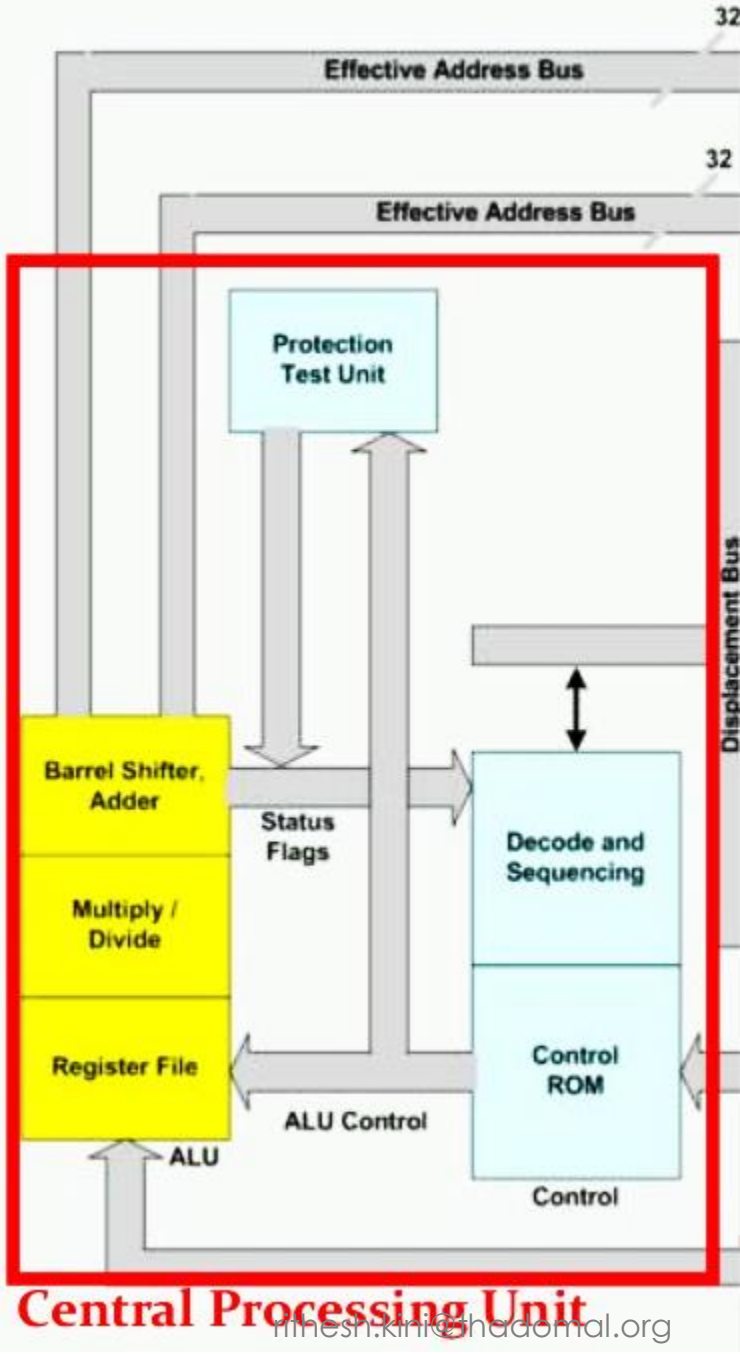


rithesh.kini@nadomal.org

Execution Unit

- **2. Data unit:**
  - The data unit contains the 32-bit ALU, eight 32-bit general purpose registers and a 64-bit barrel shifter.
  - The barrel shifter is used for multiple bit shift in one clock.
  - It increases the speed of all shift and rotate operations.
- **3. Protection test unit**

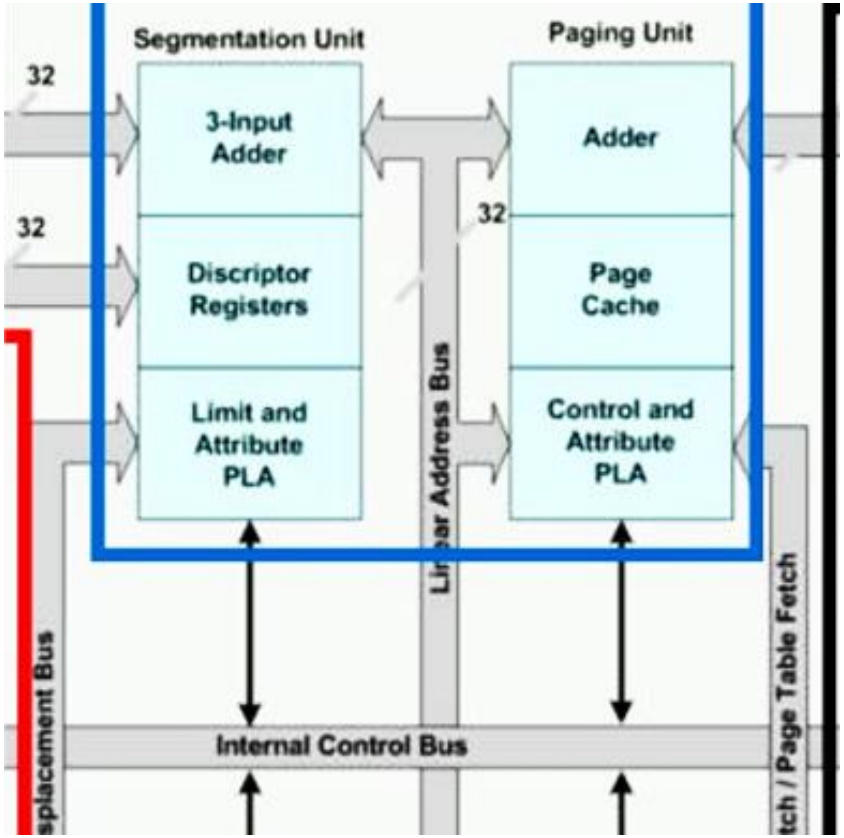
  Protection test unit checks for segmentation violations under the control of the microcode.



ritnesh.kini@thadomal.org

Memory Management Unit

- The **Memory management unit** consists of a Segmentation unit and a Paging unit.
- **Segmentation unit** allows the use of two address components, viz. segment and offset for relocatability and sharing of code and data.
- Segmentation unit allows segments of size 4Gbytes at max.
- The Segmentation unit provides a 4 level protection mechanism for protecting and isolating the system code and data from those of the application program.
- The limit and attribute PLA checks segment limits and attributes at segment level to avoid invalid accesses to code and data in the memory segments.

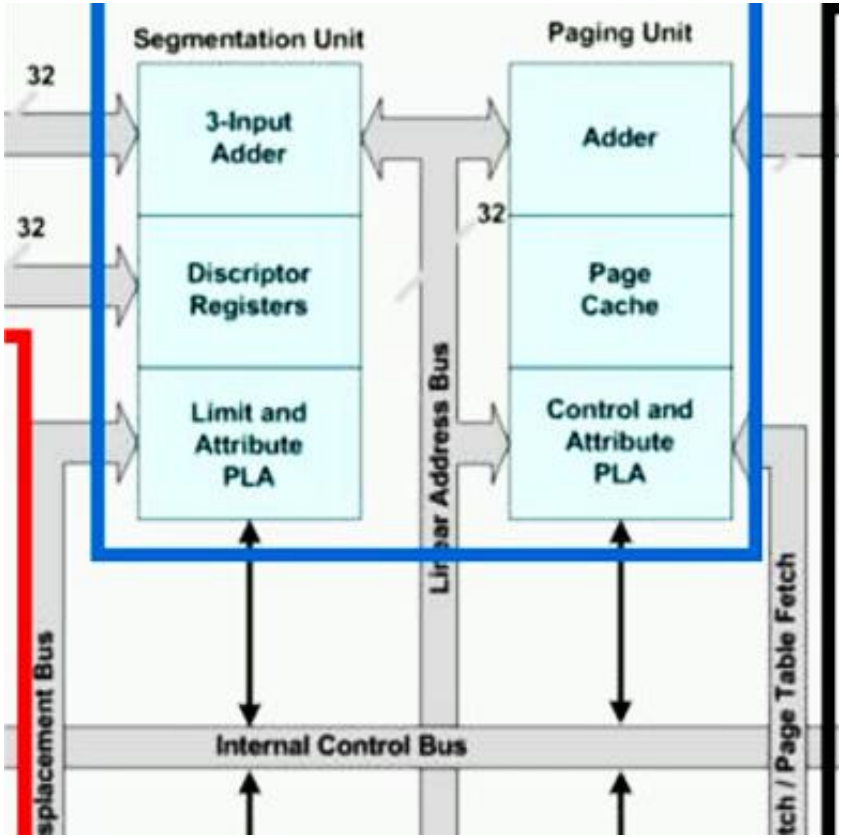


Programmable Logic Array

rithesh.kini@thadomal.org

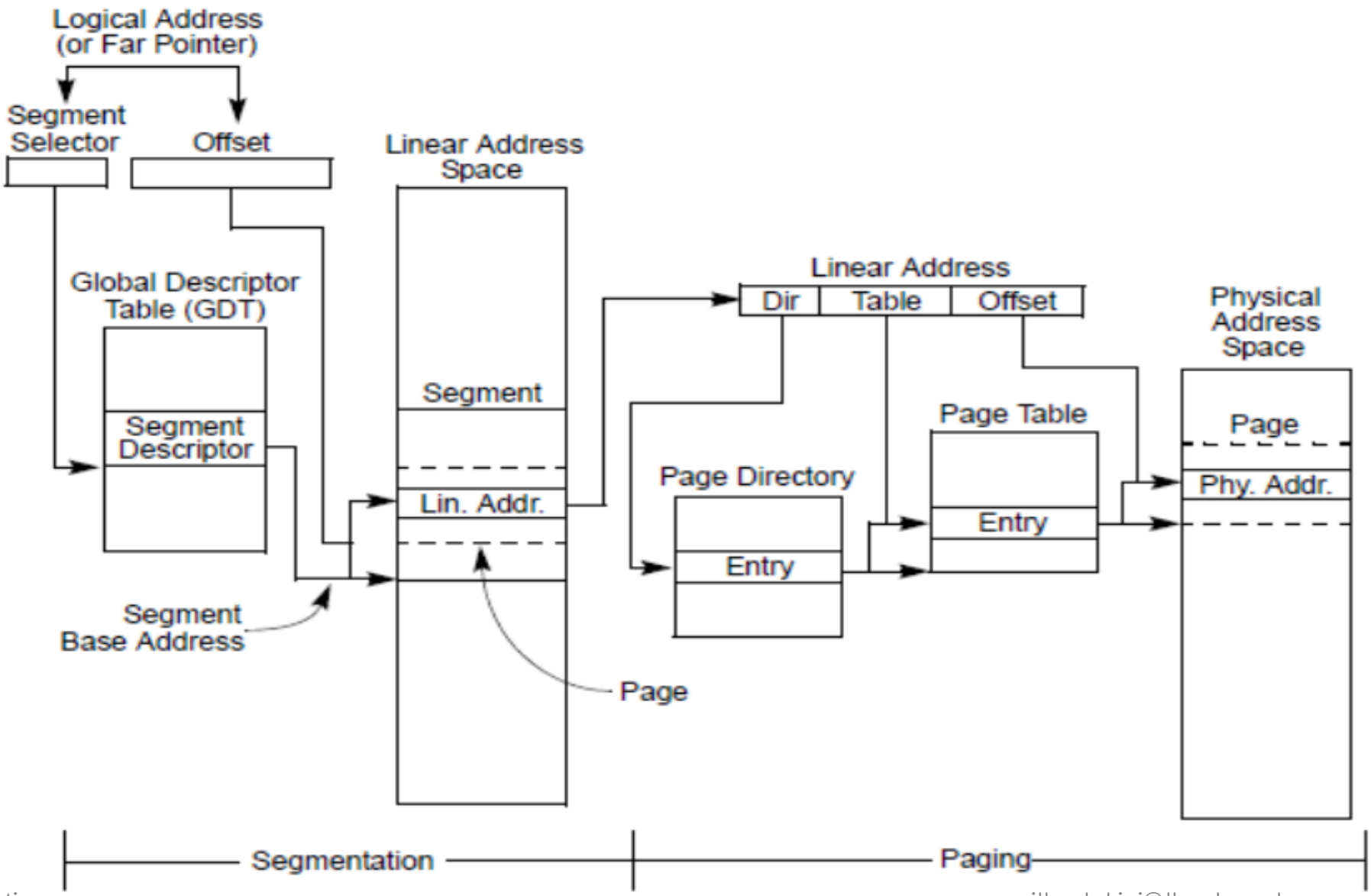
Memory Management Unit

- **Paging unit** organizes the physical memory in terms of pages of 4kbytes size each.
- Paging unit works under the control of the segmentation unit, i.e. each segment is further divided into pages.
- Paging unit converts **linear addresses into physical addresses**.
- The control and attribute PLA checks the privileges at the page level. Each of the pages maintains the paging information of the task.



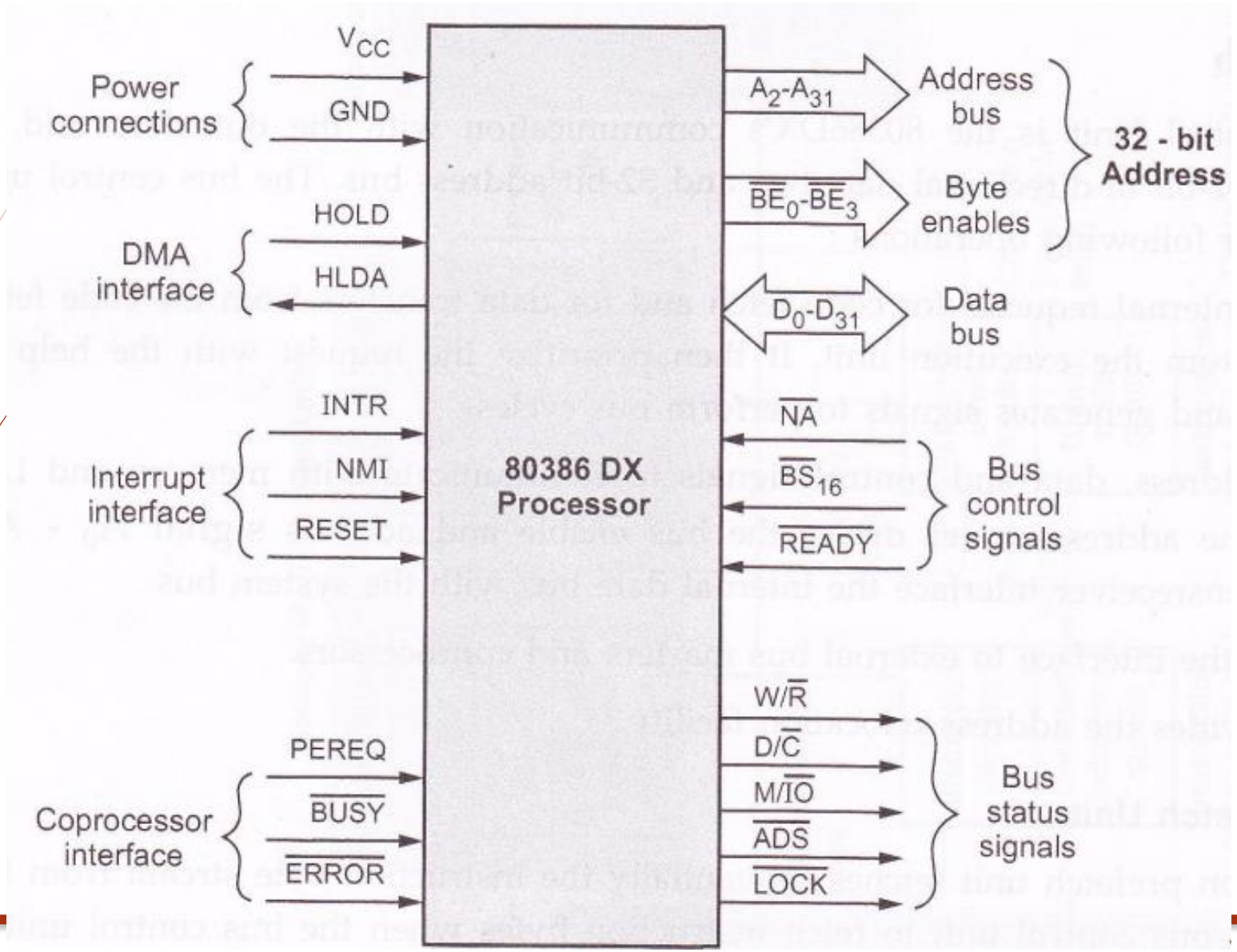
Programmable Logic Array

rithesh.kini@thadomal.org



rithesh.kini@thadomal.org

PIN DIAGRAM



rithesh.kini@thadomal.org

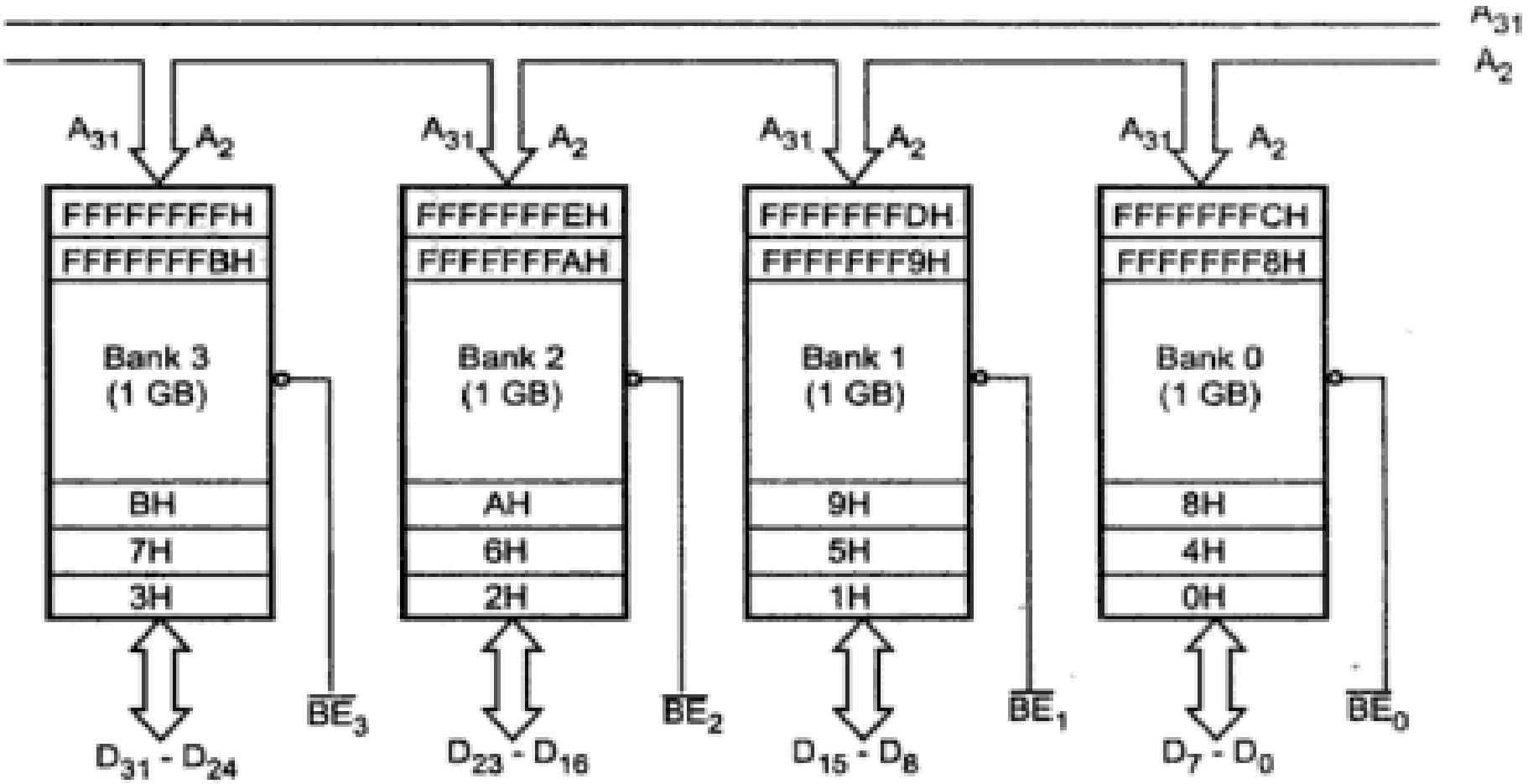
PIN DIAGRAM

- CLK2 The input pin provides the basic **system clock** timing for the operation of 80386
- D31-D0 - These 32 lines act as **bidirectional data bus** during different access cycles
- A31-A2 - **Address bus** outputs physical memory or port I/O addresses.
- BE0-BE3 –
  - **BYTE ENABLES** indicate which data bytes of the data bus take part in a bus cycle. The 32- bit data bus supported by 80386 and the memory system of 80386 can be viewed as a 4- byte wide memory access mechanism. The 4 byte enable lines BE0 to BE3, may be used for enabling these 4 banks. Using these 4 enable signal lines, the CPU may transfer 1 byte / 2 / 3 / 4 byte of data simultaneously

Signal Interface

- **Byte Enable Outputs(BE0# -- BE3#)**

BE3#	BE2#	BE1#	BE0#	Operation
1	1	1	1	No Operation
1	1	1	0	Bank0 (8-bit)
1	1	0	1	Bank1 (8-bit)
1	0	1	1	Bank2 (8-bit)
0	1	1	1	Bank3 (8-bit)
1	1	0	0	Bank 0,1 (16-bit)
1	0	0	1	Bank 1,2 (16-bit)
0	0	1	1	Bank 2,3 (16-bit)
0	0	0	0	Bank 0,1,2,3 (32-bit)



rithesh.kini@thadomal.org

PIN DIAGRAM

- $\overline{W}/R$ - **WRITE/READ** is a bus cycle definition pin that distinguishes write cycles from read cycles. (1=W, 0=R)

- $D/\overline{C}$ - **DATA/CONTROL** is a bus cycle definition pin that distinguishes data cycles (either memory or I/O), from control cycles (i.e., interrupt acknowledge, halt, and instruction fetching). (1=D, 0=C)

- $M/\overline{IO}$ - **MEMORY I/O** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles. (1=M, 0=I/O)

rithesh.kini@thadomal.org

PIN DIAGRAM

- $\overline{\text{LOCK}}$ - **BUS LOCK** is a bus cycle definition pin that enables the CPU to prevent the other bus masters from gaining the control of the system bus.
- $\overline{\text{ADS}}$ - **ADDRESS STATUS** indicates that a valid bus cycle definition and address are being driven at the Intel386 DX pins.
- $\overline{\text{NA}}$ - **NEXT ADDRESS** is used to request address pipelining.
- $\overline{\text{READY}}$ – The ready signals indicates to the CPU that the previous bus cycle has been terminated and the bus is ready for the next cycle.

rithesh.kini@thadomal.org

PIN DIAGRAM

- $\overline{\text{BS16}}$ – (**Dynamic data bus sizing**) If this pin is initialized with 0, 80386 uses 16-bit data bus and if it is initialized with 1, 80386 uses 32-bit data bus.
- HOLD - **HOLD REQUEST** input allows another bus master to request control of the local bus.
- HLDA - **HOLD ACKNOWLEDGE** output indicates that the Intel386 DX has surrendered control of its local bus to another bus master.

PIN DIAGRAM

- $\overline{\text{BUSY}}$ - signals a busy condition from a processor extension(co-processor).The busy input signal indicates to the CPU that the coprocessor is busy with the allocated task.
- $\overline{\text{ERROR}}$ - The error input pin indicates to the CPU that the coprocessor has encountered an error while executing its instruction.
- PEREQ - PROCESSOR EXTENSION REQUEST. The processor extension request output signal indicates to the CPU to fetch a data word for the coprocessor.

PIN DIAGRAM

- INTR - **INTERRUPT REQUEST** is a maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
- NMI - a **Non-Maskable Interrupt** input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
- RESET - suspends any operation in progress and places the Intel386 DX in a known reset state.

rithesh.kini@thadomal.org

PROGRAMMING MODEL

- Register Organization
  - General Purpose Registers
  - Segment Registers
  - Instruction Pointer and Flags
  - System Address Registers
  - Control Registers
  - Debug Registers
  - Test Registers.

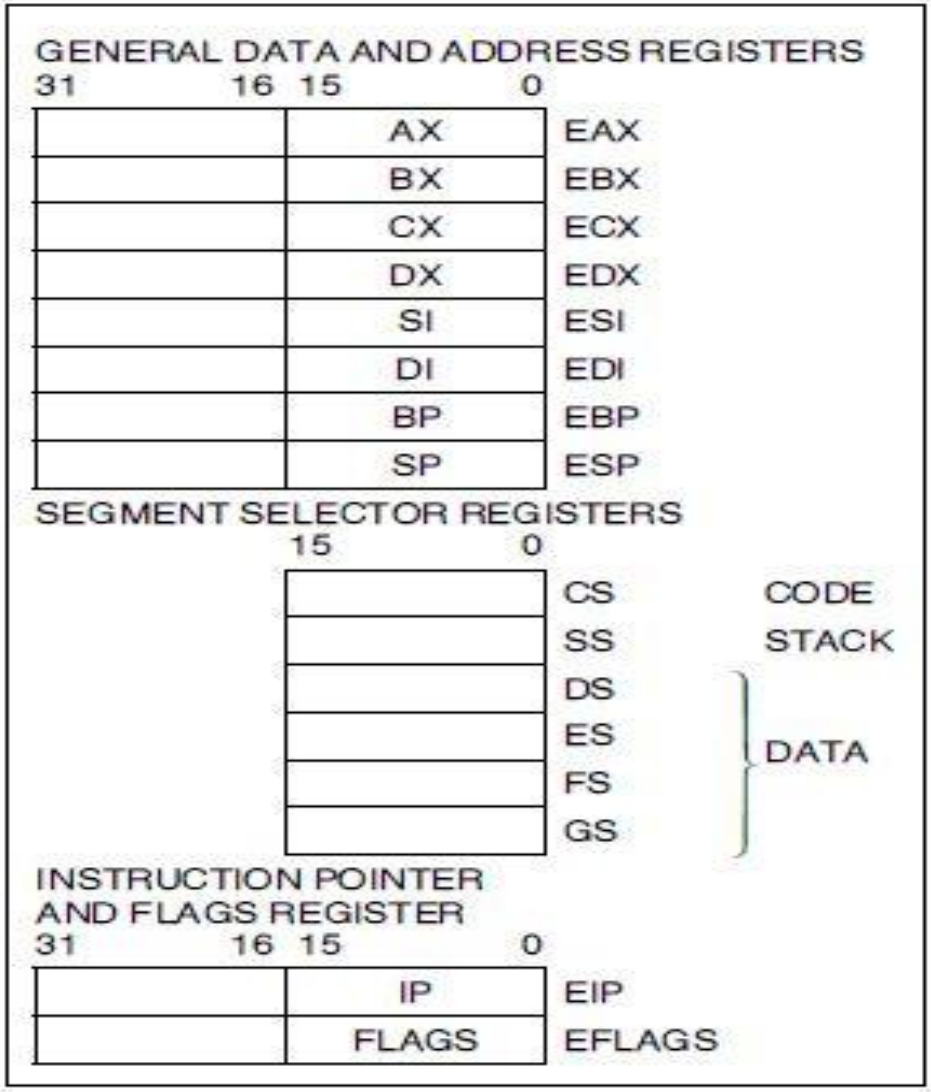
rithesh.kini@thadomal.org

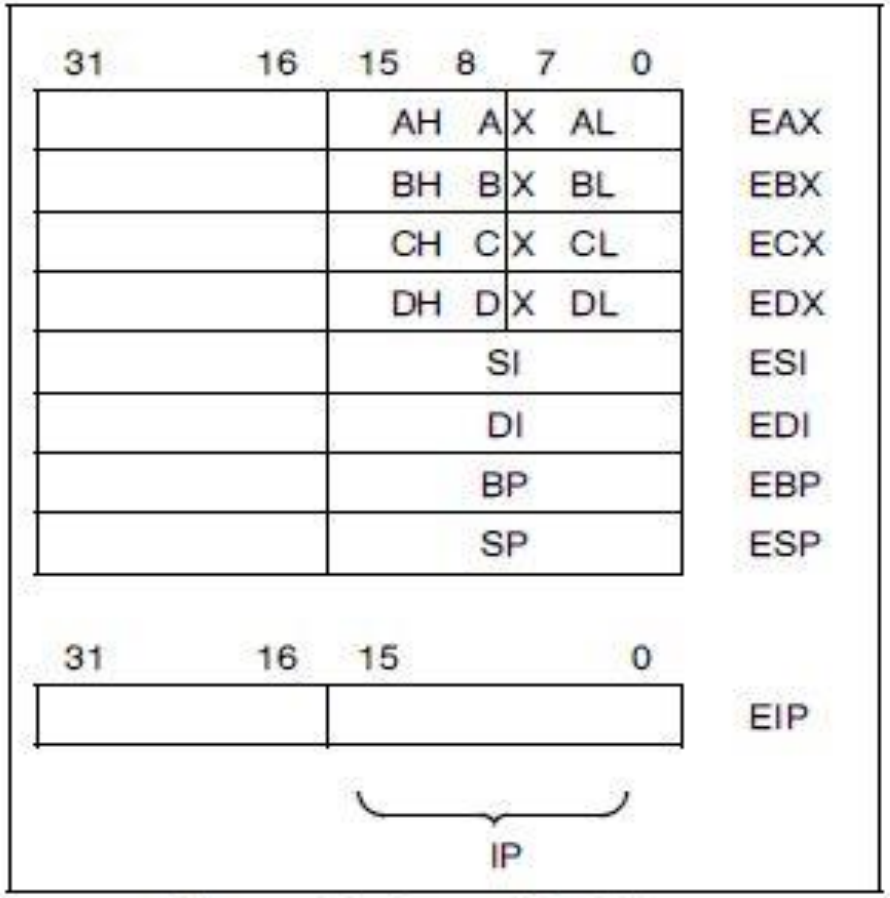
General Purpose Registers

- The eight general purpose registers of 32 bits hold data or address quantities.
- The general registers support data operands of 8, 16, and 32 bits
- The 32-bit registers are named EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP.

rithesh.kini@thadomal.org

General Purpose Registers





rithesh.kini@thadomal.org

Segment Registers

- The six segment registers available in 80386 are CS, SS, DS, ES, FS and GS.
- The CS and SS are the code and the stack segment registers respectively
- DS, ES, FS, GS are 4 data segment registers.

rithesh.kini@thadomal.org

Instruction Pointer

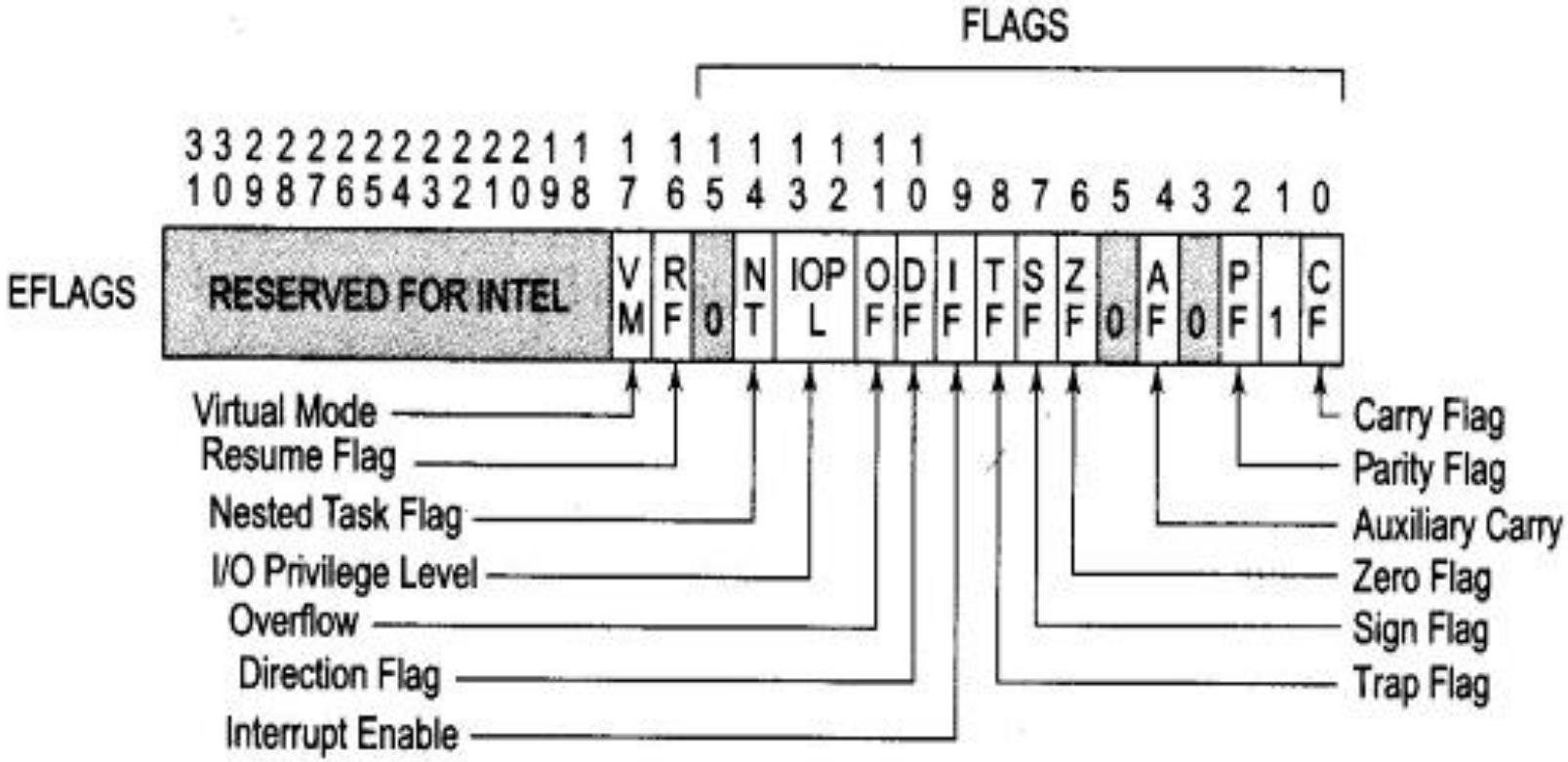
- The instruction pointer is a 32-bit register named EIP.
- EIP holds the offset of the next instruction to be executed.
- The offset is always relative to the base of the code segment (CS).
- The lower 16 bits of EIP contain the 16-bit instruction pointer named IP, which is used by 16-bit addressing.

rithesh.kini@thadomal.org

Flags Register

- The Flags Register is a 32-bit register named EFLAGS.
- The defined bits and bit fields within EFLAGS, control certain operations and indicate status of the Intel386 DX.
- The lower 16 bits of EFLAGS contain the 16-bit flag register named FLAGS, which is most useful when executing 8086 and 80286 code

EFlags Register



Note: 0 indicates Intel reserved

Flag Register of 80386 (Intel Corp.)

rithesh.kini@thadomal.org

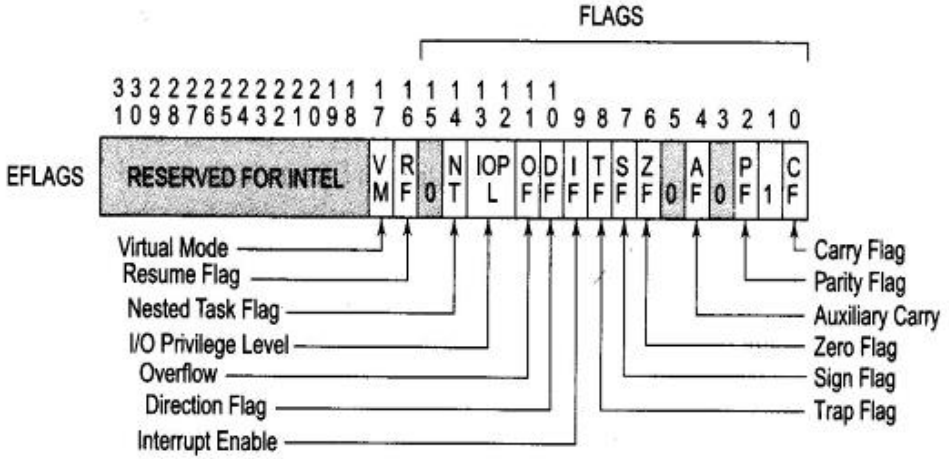
VM (Virtual 8086 Mode, bit 17)

- The VM bit provides Virtual 8086 Mode within Protected Mode.
- If set, it will switch to Virtual 8086 operation, handling segment loads as the 8086 does.
- It runs 8086 programs in a faster environment using multitasking and protection.
- The VM bit can be set only in Protected Mode, by the IRET instruction and by task switches at any privilege level.
- Once in Virtual Mode we can return back to Protected Mode by making VM← 0

rithesh.kini@thadomal.org

RF (Resume Flag, bit 16)

- The RF flag is used in conjunction with the debug register breakpoints.
- It is checked at the starting of every instruction cycle.
- When RF=1, any debug fault in the next instruction will be ignored.
- RF is then automatically reset at the successful completion of next instruction except the IRET instruction, the POPF instruction, (and JMP, CALL, and INT instructions causing a task switch).



Note: 0 indicates Intel reserved

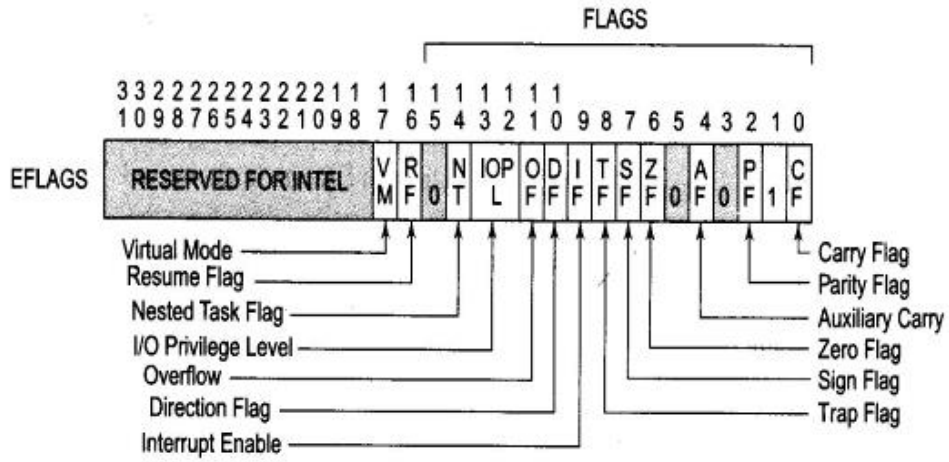
Flag Register of 80386 (Intel Corp.)

rithesh.kini@thadomal.org

NT (Nested Task, bit 14)

- This flag applies to Protected Mode.
- NT is set to indicate that the execution of this task is nested within another .
- If set, it indicates that the current nested task's Task State Segment (TSS) has a valid back link to the previous task's TSS.
- This bit is set or reset by control transfers to other tasks.
- Example:

  Call instruction



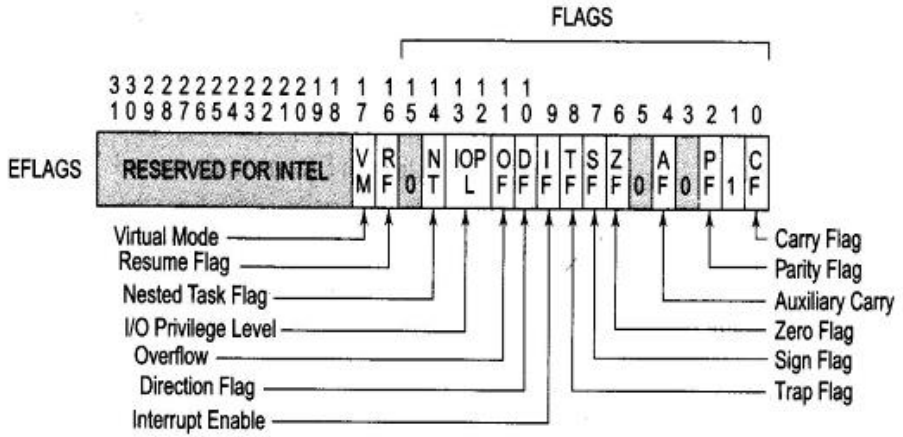
Note: 0 indicates Intel reserved

Flag Register of 80386 (Intel Corp.)

rithesh.kini@thadomal.org

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

- This two-bit field applies to Protected Mode.
- IOPL indicates the numerically maximum CPL (current privilege level) value permitted to execute I/O instructions without generating an exception
- It indicates the privilege level at which the task should be executed to access the I/O device



Note: 0 indicates Intel reserved

Flag Register of 80386 (Intel Corp.)

rithesh.kini@thadomal.org

OF (Overflow Flag, bit 11)

- OF is set if the operation resulted in a signed overflow.
- Signed overflow occurs when the operation resulted in carry/borrow into the sign bit (high-order bit) of the result but did not result in a carry/borrow out of the high order bit, or vice-versa.
- For 8/16/32 bit operations, OF is set according to overflow at bit 7/15/31, respectively.

DF (Direction Flag, bit 10)

- DF defines whether ESI and/or EDI registers post decrement or post increment during the string instructions.
- Post increment occurs if DF is reset.
- Post decrement occurs if DF is set.

IF (INTR Enable Flag, bit 9)

- The IF flag, when set, allows recognition of external interrupts signal on the INTR pin.
- When IF is reset, external interrupts signal on the INTR are not recognized.

TF (Trap Enable Flag, bit 8)

- It is used to perform single stepping
- TF=1 – perform single stepping
- TF=0 – do not perform single stepping

rithesh.kini@thadomal.org

SF (Sign Flag, bit 7)

- Most operations set this bit the same as the most significant bit (of high-order bit) of the result. 0 is positive, 1 is negative

ZF (Zero Flag, bit 6)

- ZF is set if all bits of the result are 0.
- Otherwise it is reset.

AF (Auxiliary Carry Flag, bit 4)

- It indicates a carry from lower nibble to higher nibble

PF (Parity Flags, bit 2)

- PF is set if the low-order eight bits of the operation contains an even number of ``1's'' (even parity).
- PF is reset if the low-order eight bits have odd parity.
- PF is a function of only the low-order eight bits, regardless of operand size.

CF (Carry Flag, bit 0)

- CF is set if the operation resulted in a carry out of (addition), or a borrow into (subtraction) the high-order bit.
- Otherwise CF is reset.
- For 8-, 16- or 32-bit operations, CF is set according to carry/borrow at bit 7, 15 or 31, respectively.
- Note in these descriptions, ``set'' means ``set to 1,'' and ``reset'' means ``reset to 0.''

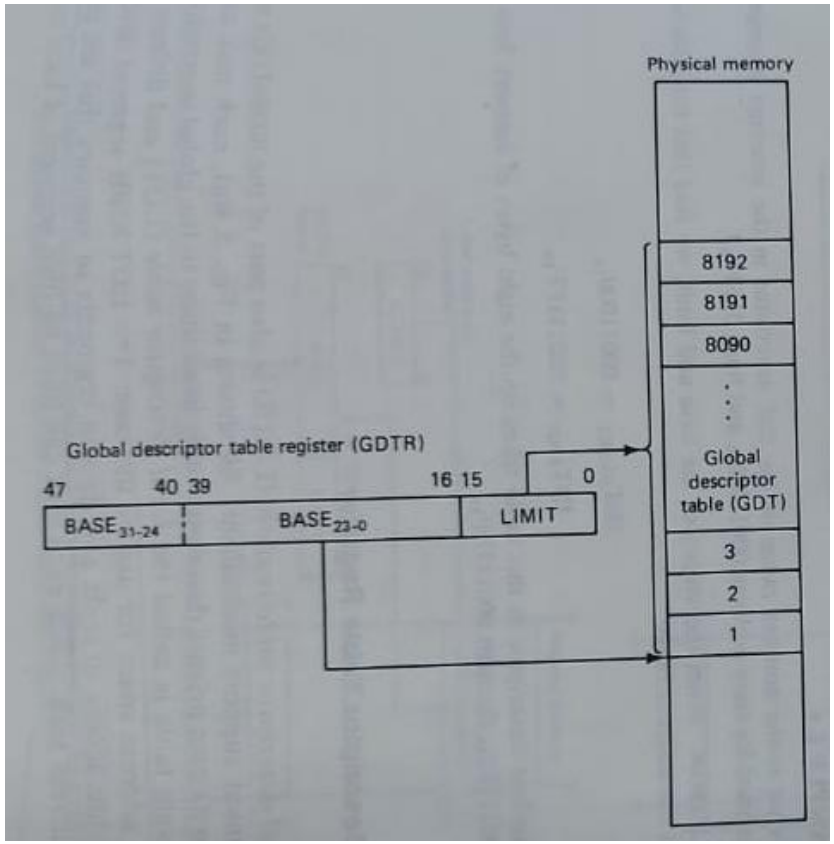
System Address Registers/Descriptor tables

- There are four new registers in the protected mode model:
  - Global Descriptor Table Register (GDTR)
  - Interrupt Descriptor Table Register (IDTR)
  - Local Descriptor Table Register (LDTR)
  - Task State Segment Descriptor Register (TR)

rithesh.kini@thadomal.org

Global Descriptor Table Register-GDTR

- The contents of the **GDTR reg** define a **table** in the 80386DX's physical memory address space called the **Global Descriptor Table**
- GDTR is a **48-bit** register to address GDT.
- The **lower 2 bytes** of the register, identified as the **LIMIT**, specify the size in bytes of the GDT. (max size = 65,536 bytes))
- The **upper 4 bytes** of the GDTR labeled as the **BASE**, locate the beginning of the GDT in physical memory.
- Storage locations in the **global memory** is accessible by **any task** that runs on the microprocessor.
- Only 1 GDT exists for all software tasks.
- Each **system segment descriptor** in the table is **8 byte** long :
  - Identify characteristics of the segments of global memory.
  - i.e, size, starting point, access rights of Global memory segment.

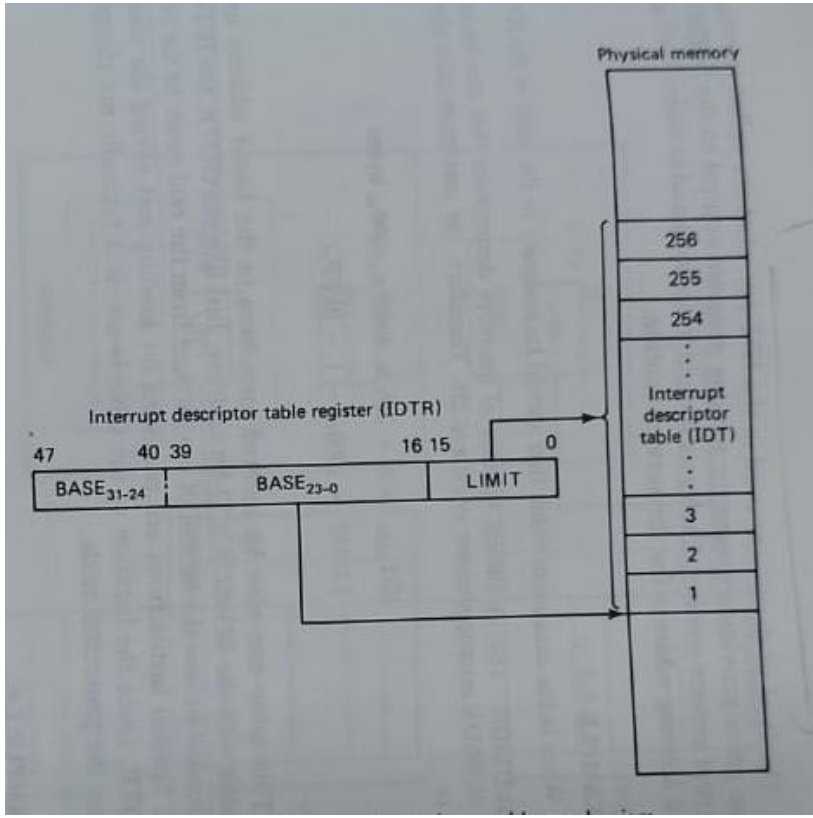


Limit = $00FF_{16}$ => 32 descriptors (256/8)

rithesh.kini@thadomal.org

Interrupt Descriptor Table Register

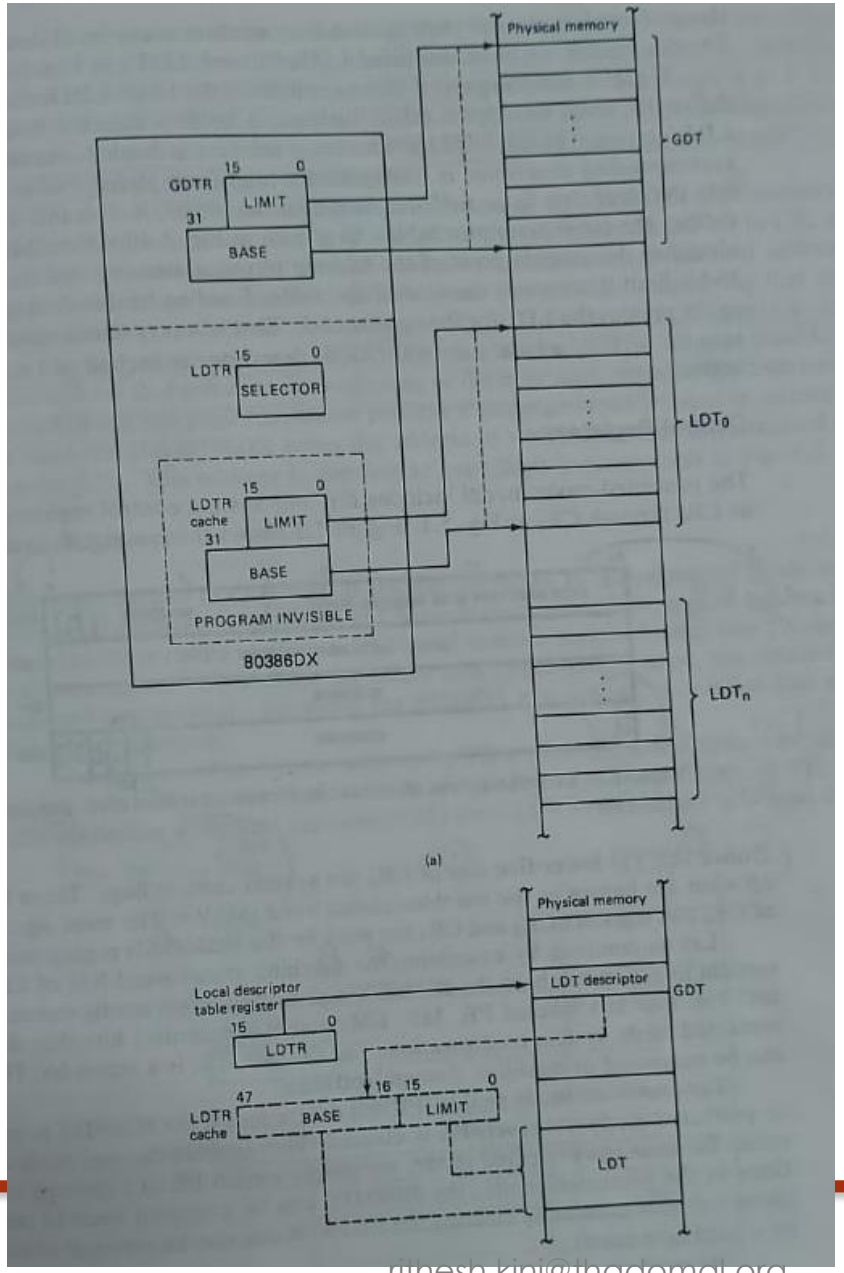
- Just like GDTR, the **IDTR** defines a table in the physical memory.
- This register and table provide the mechanism by which the microprocessor passes the program control to the **interrupt and the exception service routines.**
- IDTR also is of **48 bits** in length.
- Again, the **lower two bytes** define the size, and it also can be up to 65,536 bytes long.
- But 386 supports up to **256** interrupts and exceptions.
- The **upper 4 bytes** identify the base address of the IDT in physical memory.
- Just like GDTR, IDTR needs to be loaded prior to switching to the protected mode.



rithesh.kini@thadomal.org

Local Descriptor Table Register

- **LDTR** is a **16 bit register**.
- It holds a **selector** that **points to LDT Descriptor** in GDT
- **LDT Descriptor**
  - (32 bits – Base address of LDT, 16 Bits – Limit, access rights)
  - Each task can have access to its own **private** descriptor table in addition to the GDT called LDT (Fig).
  - Defines a local memory address space for use by the task.
  - Provide to access to code and data reserved for the current task.
  - Since each task can have its own segment of local memory, the protected mode can contain many LDT (LDT_0 to LDT_n).



nilesh.kini@indodmai.org

Task Register

- It is a **16 bit register**
- The **selector** in TR is used to locate a descriptor in the GDT.
- This descriptor defines a block of memory called the **Task state segment(TSS).**
- It does this by providing the starting address (BASE) and the size-LIMIT of the segment.
- Every task has its own TSS (task State segment)
- The TSS holds the information needed to initiate a task, such as initial values for the user accessible registers, a back link to the previous task

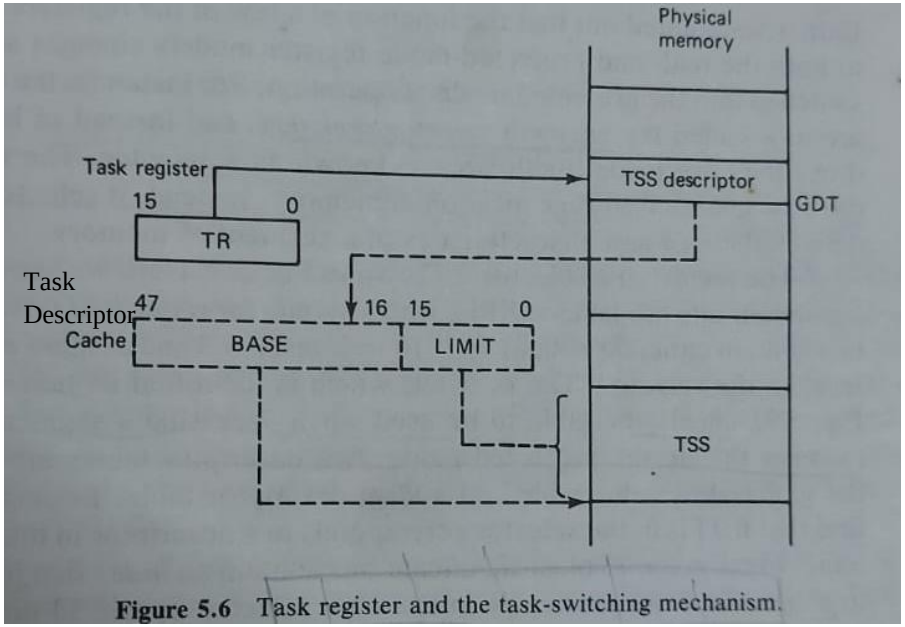


Figure 5.6 Task register and the task-switching mechanism.

rithesh.kini@thadomal.org

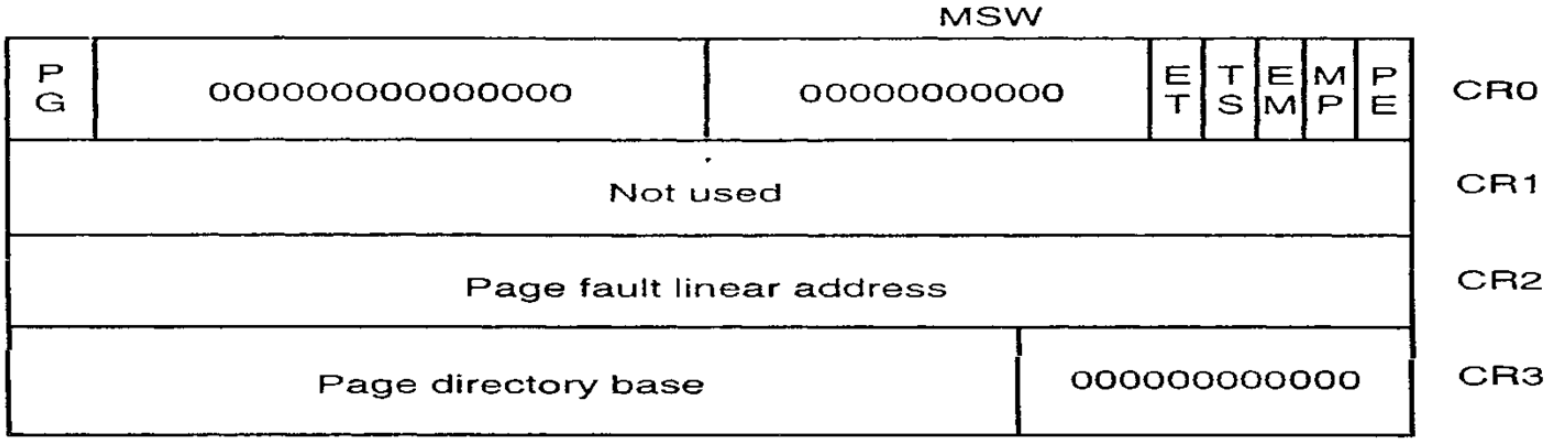
Control Registers

- The protected mode includes the 4 system control registers, identified as CR0 to CR3.
- These are 32 bit registers.
- The lower **5 bits** of the **CR0** are system **control flags**.
- These bits make up what is known as the **machine status word-MSW.**
- MSW bits of the CR0 contain
  - PE, MP, EM, and ET: control bits
  - TS: status bit.

CR0

- PG - Paging Enable *(to turn on paging)*
- Machine Status Word
  - PE – Protection Enable *(to select between real/protected mode)*
  - MP – Math Co-processor Present
  - EM – Emulate coprocessor
  - ET – Extension Type *(select between 80287/80387)*
  - TS – Task Switched

CR0



Register CR0 contains a number of special control bits that are defined as follows in the 80386:

PG Selects page table translation of linear addresses into physical addresses when PG = 1. Page table translation allows any linear address to be assigned any physical memory location.

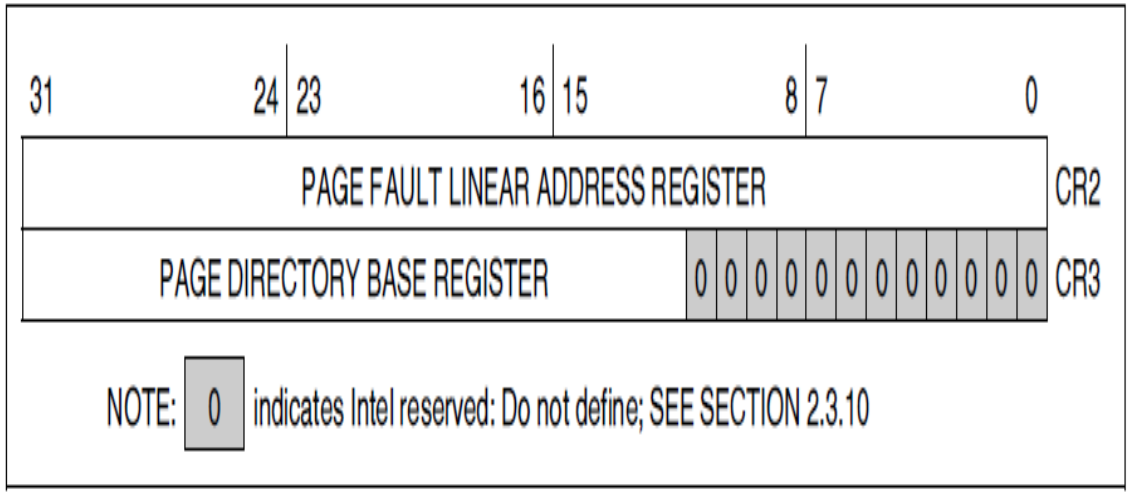
ET Selects the 80287 coprocessor when ET = 0 or the 80387 coprocessor when ET = 1. This bit was installed because there was no 80387 available when the 80386 first appeared. In most systems, ET is set to indicate that an 80387 is present in the system.

TS	Indicates that the 80386 has switched tasks (in protected mode, changing the contents of TR places a 1 into TS). If TS = 1, a numeric coprocessor instruction causes a type 7 (coprocessor not available) interrupt.
EM	Is set to cause a type 7 interrupt for each ESC instruction. (ESCape instructions are used to encode instructions for the 80387 coprocessor.) We often use this interrupt to emulate, with software, the function of the coprocessor. Emulation reduces the system cost, but it often takes at least 100 times longer to execute the emulated coprocessor instructions.
MP	Is set to indicate that the arithmetic coprocessor is present in the system.
PE	Is set to select the protected mode of operation for the 80386. It may also be cleared to reenter the real mode. This bit can only be set in the 80286. The 80286 could not return to real mode without a hardware reset, which precludes its use in most systems that use protected mode.

CR1 and CR2

- CR1: reserved
  - CR1 is reserved for use in future Intel processors.
- CR2: Page Fault Linear Address
  - holds the 32-bit linear address that caused the last page fault detected.

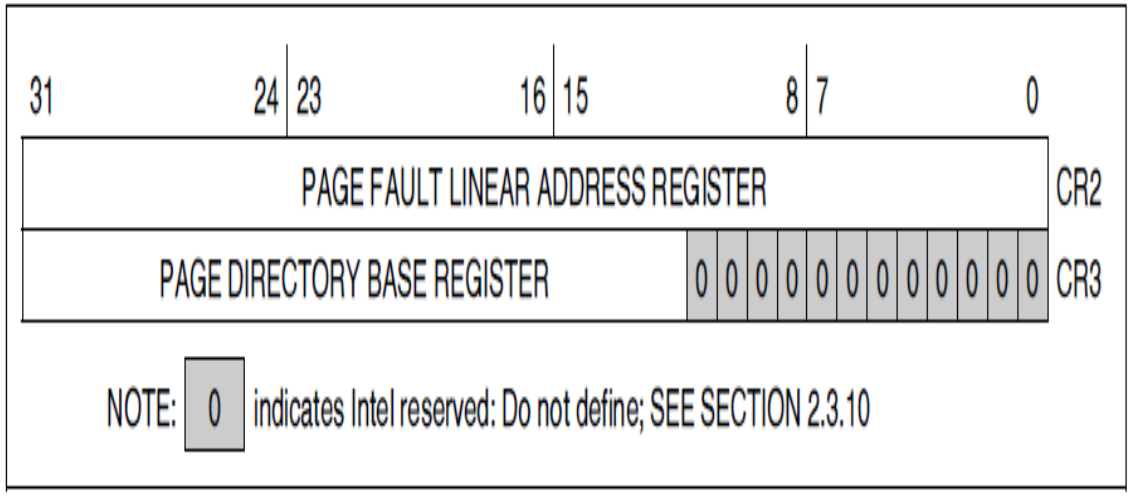
(A page fault occurs when a desired page in not present in the physical memory during page translation process)



rithesh.kini@thadomal.org

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). Therefore the lowest twelve bits of CR3 are ignored when written and they store as undefined.
- It gives upper 20 bits of the starting address of the page directory



rithesh.kini@thadomal.org

Debug and Test Registers

- Debug Registers
  - 8 debug registers for hardware debugging DR0 – DR7
  - DR4, DR5 – Intel reserved
  - DR0- DR3 – store four program controllable breakpoint addresses
  - DR6, DR7 – hold breakpoint status, breakpoint control information
- Test Registers
  - Test control
  - Test status registers

Debug Registers

DEBUG REGISTERS

31 … 0	
Linear Breakpoint Address 0	DR_0
Linear Breakpoint Address 1	DR_1
Linear Breakpoint Address 2	DR_2
Linear Breakpoint Address 3	DR_3
Intel Reserved.	DR_4
Intel Reserved.	DR_5
Breakpoint Status	DR_6
Breakpoint Control	DR_7

Test Register (for Page Cache)

31 … 0	
Test Control	TR_6
Test Status	TR_7

Debug and Test Registers of 80386 (Intel Corp.)

rithesh.kini@thadomal.org

rithesh.kini@thadomal.org