

The 80386-80486  
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**
- **Although it had long been obsolete as a personal computer CPU, Intel and others had continued making the chip for embedded systems.**
- **Used in aerospace technology**

# 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,32-bit internal registers and 24-bit external address bus.

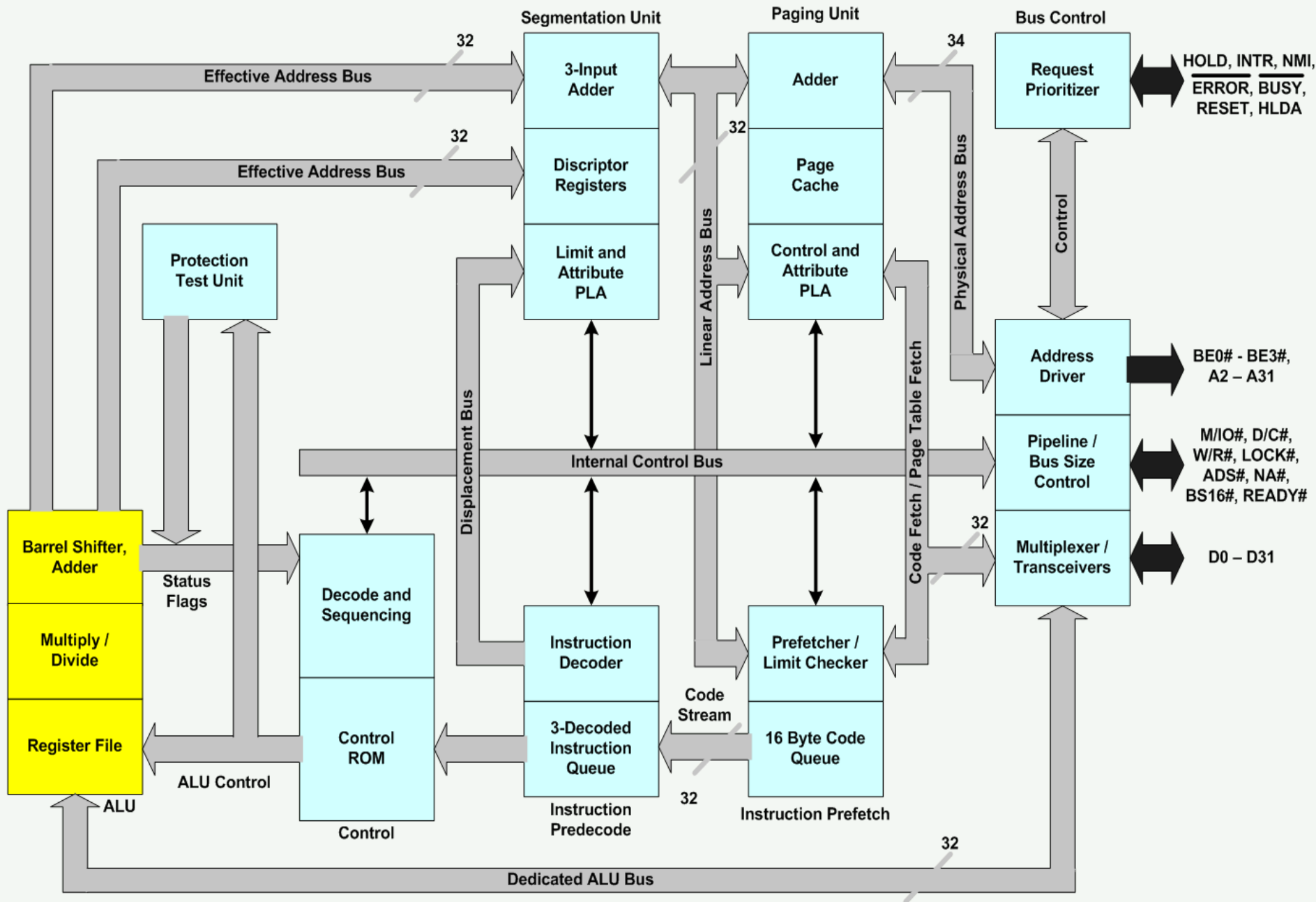
## **80386SL** –

- low-power microprocessor with power management features, with 16-bit external data bus and 24-bit external address bus.
- The processor included ISA bus controller, memory controller and cache controller.

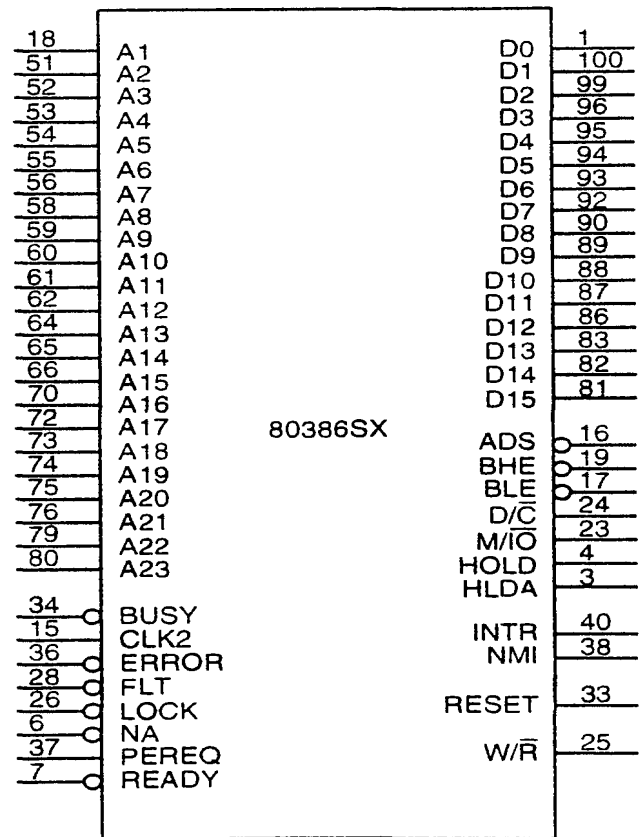
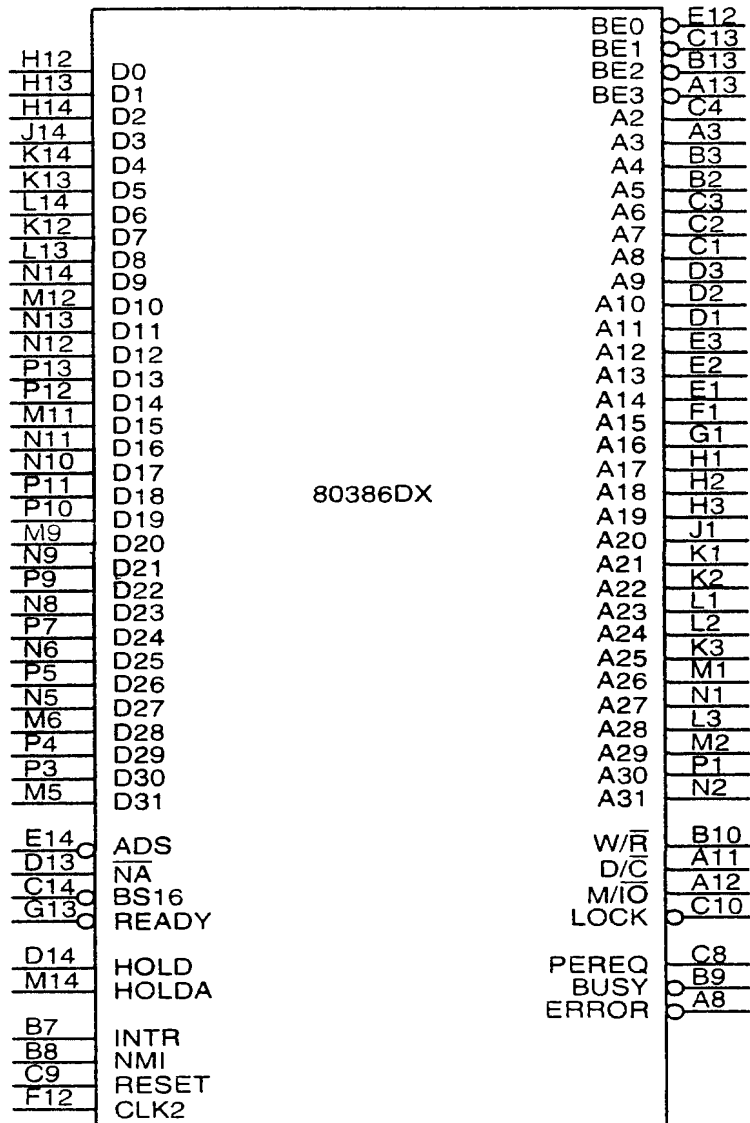
**Embedded 80376 and 80386EX processors.**- Still in use today.

# *Internal Architecture*

- To enhance performance 80386 has 6 functional units, processing in parallel:
  - The bus unit
  - The prefetch unit
  - The decode unit
  - The execution unit
  - The page unit
  - The segment unit



PLA: Programmable Logic Array



# The Bus Unit

- The bus unit is the interface to the external devices.
- The bus interface unit provides a 32-bit data bus, a 3-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.
- 80386 has separate pins for its address and data bus lines.
- This processing unit contains the latches and drivers for the address bus, transceivers for the data bus, and control logic for signaling whether a memory input/output, or interrupt-acknowledgement bus cycle is to be performed.

# The Prefetch Unit

- The prefetch unit performs a mechanism known as an *instruction stream queue*.
- This queue permits a prefetch upto 16 bytes of instruction code.
- Whenever the queue is not full, and the execution unit is not asking the bus unit to read or write data from the memory, the prefetch queue supplies addresses to the bus interface unit and signals it to look ahead in the program by fetching the next sequential instructions.
- Prefetched instructions are held in the FIFO queue for use by the instruction decoder.
- Whenever bytes are loaded into the input end of the queue, they are automatically shifted up through the FIFO to the empty location near the output



# The Decode Unit

- The decode unit accesses the output end of the prefetch unit's instruction queue.
- It reads the machine-code instructions from the output side of the prefetch queue and decodes them into microcode instruction format used by the execution unit, thus it off-loads the responsibility for the instruction decoding from the execution unit.
- The instruction queue, a part of the decode unit permits three fully decoded instructions to be held waiting for use by the execution unit.
- Thus it improves the performance of the CPU.

# The Execution Unit

- The execution unit involves the arithmetic/logic unit-ALU, registers, special multiply, divide, and shift hardware, and a control ROM.
- The control ROM contains the microcode sequences that define the operation performed by each of the machine code instructions.
- The execution unit reads the decoded instructions from the instruction queue and performs the operations that are specified.
- During the execution of an instruction, it requests the segment and page units to generate operand addresses and the bus interface unit to perform read or write bus cycles to access data in memory or I/O devices

# The Page and Segment Unit

- The segment and the Page units provide the memory management and protection services for the 80386.
- They offload the responsibility for address generation, address translation and segment checking from the bus interface unit and thereby further boosting the performance of the CPU.
- The segment unit implements the segmentation model of the 386 memory management.
- i.e. It contains dedicated hardware for performing high speed address calculations, logical to linear address translation and protection checks.

- The page unit implements the protected mode paging model of the 80386's memory management.
- It contains the translation look aside buffer that stores recently used page directory and page table entries.
- When paging is enabled, the linear address produced by the segment unit is used as the input to the paging unit.
- Here the linear address is translated into the physical address of the memory or I/O location to be accessed. Thus physical memory is the output to the bus interface unit.

# *Real Address Mode Software*

## *Model of the 80386DX*

- Just like 80286, 80386DX comes up in the real address mode after it is reset
- The CPU will remain in this mode unless it is switched to protected mode by the software
- When in the real mode, the 80386DX can be used to execute the base instruction set of the 8086/8088 architecture.
- Similar to 80286, object code for the base instructions of the 80386 is identical to that of the 8086/8088. Thus code compatibility is maintained between them.
- The real mode of the 386 generates the physical addresses in the same way as in the 8086 or 80286. That is 16 bit segment register contents and the 16 bit offset are used to generate the absolute 20 bit physical address.
- Note: IP has 32 bits but in real mode only lower 16 bits are active.

# REAL MODE SOFTWARE MODEL OF 80386DX

- The register model of 80386dx is quite different from those of the 8088, 8086 and 80286.
- There are 17 internal registers that are used in the real mode application programming.
- Nine of them –the data registers-(EAX, EBX, ECX and EDX), the pointer registers (EBP and ESP), the index registers-(ESI and EDI) and the flag register (EFLAG) - are identical to the corresponding registers in the 8086's software model except that they are now 32 bits in length.
- On the other hand the segment registers (CS, DS, ES and SS) and the instruction pointers (IP) are both identical and still 16 bits in length.
- Several new registers are found in the real-mode 80386DX's software model. For instance, it has two more data segment registers, FS and GS.

- Another new register is called the Control Register Zero (CRo).
- The five least significant bits of this register are called the machine status word-MSW and are identical to the MSW of the 80286.
- The only bit in CRo that is active in the real mode is bit 0, which is the protection enable (PE) bit.
- PE is the bit used to switch the 80386dx from the real mode to the protected mode.
- At reset, PE is set to Zero and thus Real mode is selected at reset.

# 80386 Specific Instruction Set:

- The base instruction set was enhanced in the 80286 microprocessor with a group of instructions known as the extended instruction set.
- All these instructions are also available in the 80386 real mode.
- The enhancement to the 80386dx's real mode instruction set is the 80386 specific instruction set.
- It includes instructions to directly load a pointer into the FS, GS and SS registers.
- A number of special purpose instructions have been added in the instruction set of 386.



# *Protected Address mode Software Architecture of 80386dx*

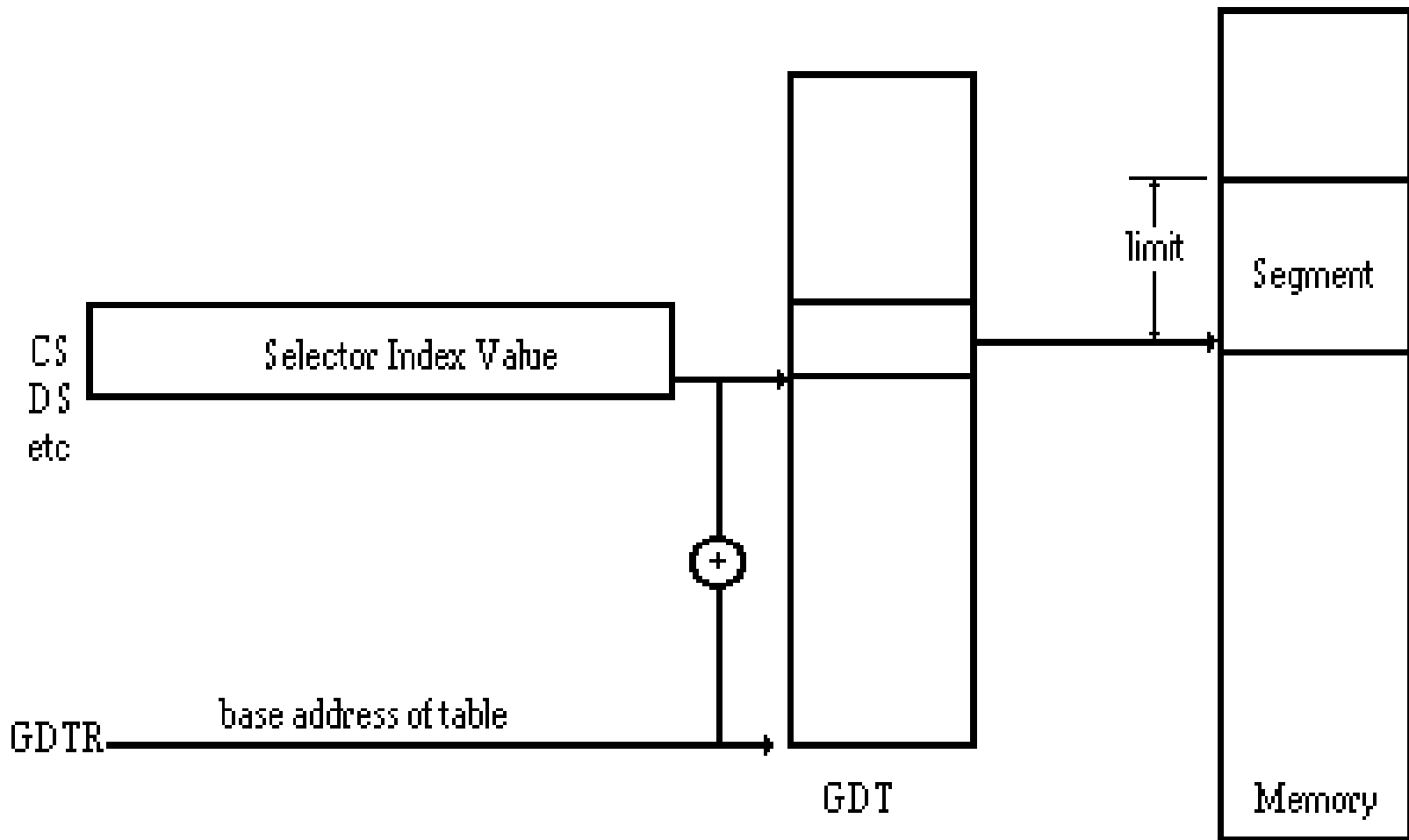
- When configured for the protected mode operation, the 80386dx microprocessor provides an advanced software architecture that supports memory management, virtual addressing, paging and multitasking. There are four new registers in the protected mode model:
- *The Global Descriptor Table Register(GDTR)*
- *Interrupt Descriptor Table Register(IDTR),*
- *Local Descriptor Table Register(LDTR)*
- *Task Register (TR)*

# 1. The Global Descriptor Table

## Register-GDTR

- The contents of the GDTR define a table in the 80386dx's physical memory address space called the Global Descriptor Table, which is one important element in the CPU's memory management system.
- GDTR is a 48-bit register located inside the 80386DX to address GDT.
- The lower 2 bytes of the register, identified as the LIMIT, specify the size in bytes of the GDT.
- The upper 4 bytes of the GDTR labeled as the BASE, locate the beginning of the GDT in physical memory.
- This 32 bit address allows the table to be positioned anywhere in the 4 Gbyte linear address space.
- The GDT provides a mechanism for defining the characteristics of the 80386's global memory address space. .
- i.e. Storage locations in the global memory is accessible by any task that runs on the microprocessor.
- Only 1 GDT exists for all programs.

- This table contains what are called *System segment Descriptors*.
- These descriptors identify the characteristics of the segments of the global memory.
- For instance, a segment descriptor provides information about the size, starting point, and access rights of a global memory segment.
- Each descriptor is 8 bytes long. GDT can hold maximum up to 8192 descriptors.
- The value of BASE and LIMIT must be loaded into the GDTR before the processor is switched from the real mode of operation to the protected mode.
- Once in protected mode, the location of the table is not changed.



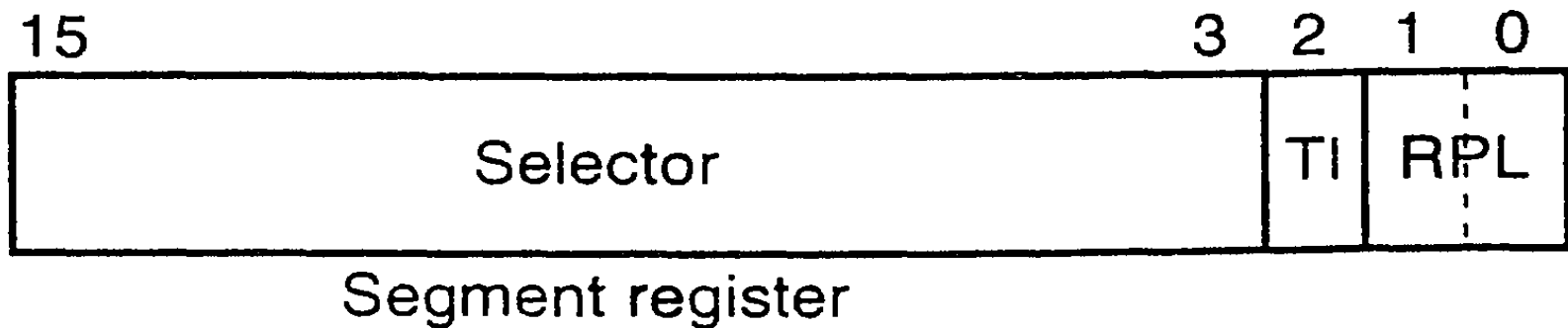
# *2. Interrupt Descriptor Table* *Register*

- Just like GDTR, the IDTR defines a table in the physical memory.
- The contents of the table are Interrupt Descriptors, not segment descriptors.
- 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 starting address of the IDT in physical memory.
- IDTR needs to be loaded prior to switching to the protected mode.

## *3. Local Descriptor Table Register*

- LDTR is also a part of the memory management support mechanism.
- Each task can have access to its own private descriptor table in addition to the GDT.
- The private descriptor table is called LDT and defines a local memory address space for use by the task.
- The LDT holds segment descriptors that provide access to the codes and the data in segments of memory that are reserved for the current task.
- Since each task can have its own segment of local memory, the protected mode can contain many LDT.
- LDTR is of 16 bit.

- The contents of the 16 bit LDTR do not directly define the local descriptor table.
- Instead it holds a selector that points to an LDT descriptor in the GDT.
- Whenever a selector is loaded into the LDTR, the corresponding descriptor is apparently read from the global memory and loaded into the LDT cache within the processor.
- Every time the selector is loaded into the LDTR, a local descriptor is cached and a new LDT is activated.



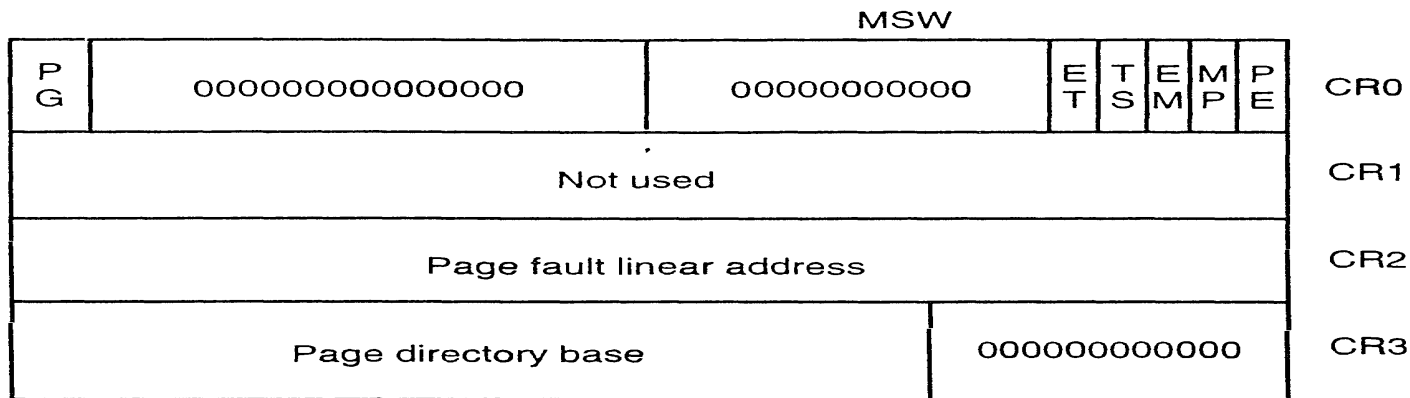
Whenever a new selector is placed into one of the segment registers, the 80386 accesses one of the descriptor tables and automatically loads the descriptor into a program-invisible cache portion of the segment register. As long as the selector remains the same in the segment register, no additional accesses are required to the descriptor table. The operation of fetching a new descriptor from the descriptor table is program-invisible because the microprocessor automatically accomplishes this each time that the segment register contents are changed in the protected mode.



- In selector format, the two least significant bits are RPL- Requested privilege level, which assigns a privilege level to the selector.
- The next bit is identified as task indicator-TI, which selects the table to be used when accessing a segment descriptor.
- If TI is 0, the selector corresponds to a descriptor in the GDT else LDT.
- The 13 most significant bits contain an index that is used as a pointer to a specific descriptor entry in the table selected by the TI bit.

## 4. 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.
- The most significant bit of the CR0 and registers CR2 and CR3 are used by the 80386's paging mechanism
- MSW bits of the CR0 contain PE, MP, EM, and R control bits which define the protected mode system configuration and status.
- The fifth bit TS, is a status bit.



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

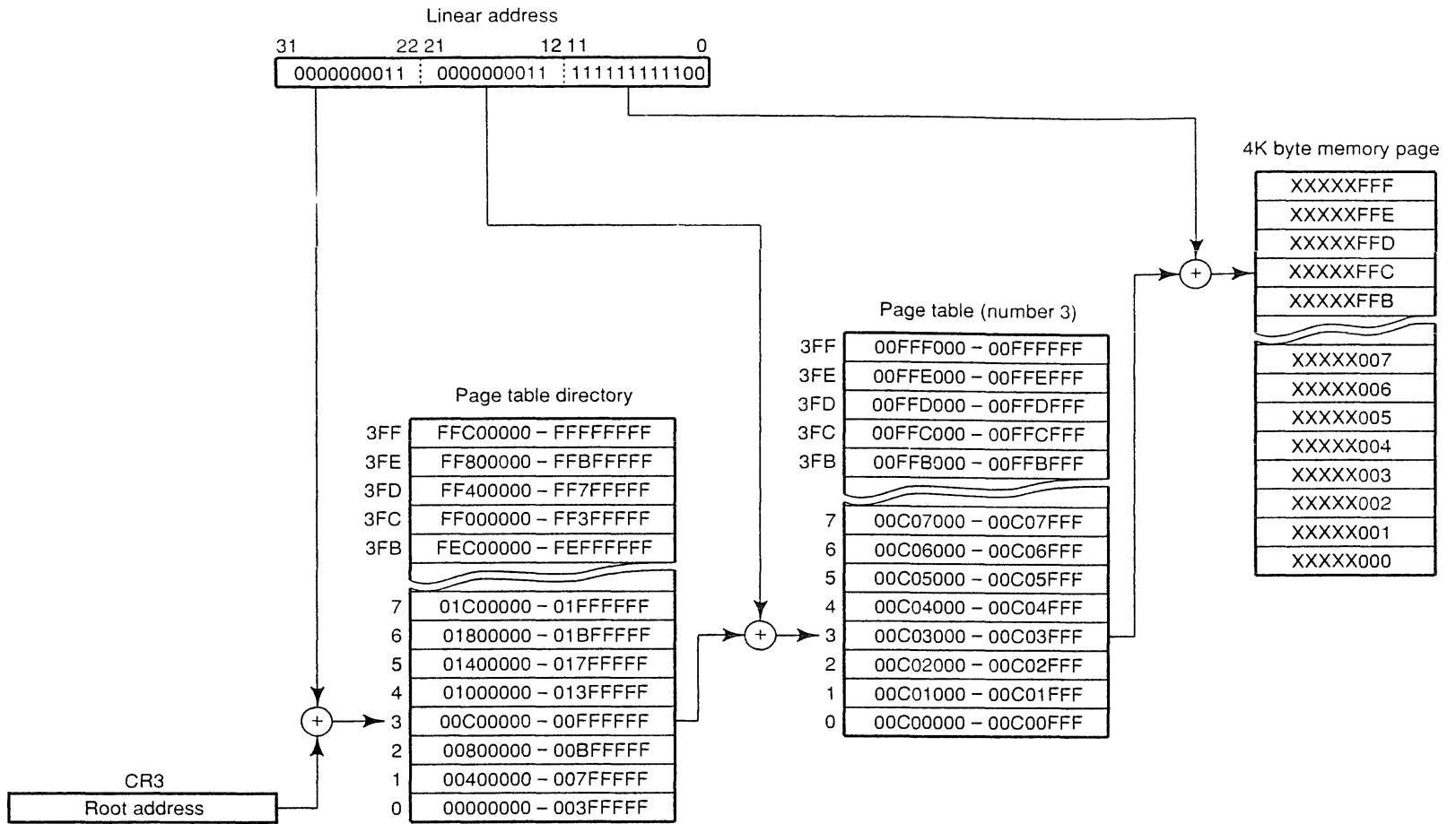
**PG** Selects page table translation of linear addresses into physical addresses when PG = 1. Page table translation allows any linear address to be assigned any physical memory location.

**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.

# Paging

- The Protected mode also supports the paged memory operation.
- Switching the PG bit in the CR0 to 1 turns on the paging.
- Now addressing of physical memory is implemented with an address translation mechanism that consists of a page directory and page table, which are both held in the physical memory.
- Control Register CR3 contains the page directory base register – PDBR.
- This register holds a 20 bit page directory base address which points to the beginning of the page directory.
- A page fault error occurs in the page translation process if page is not present in the memory.
- In this case, the 80386dx saves the address at which the page fault occurred in register CR2.
- This address is denoted as page fault linear address.



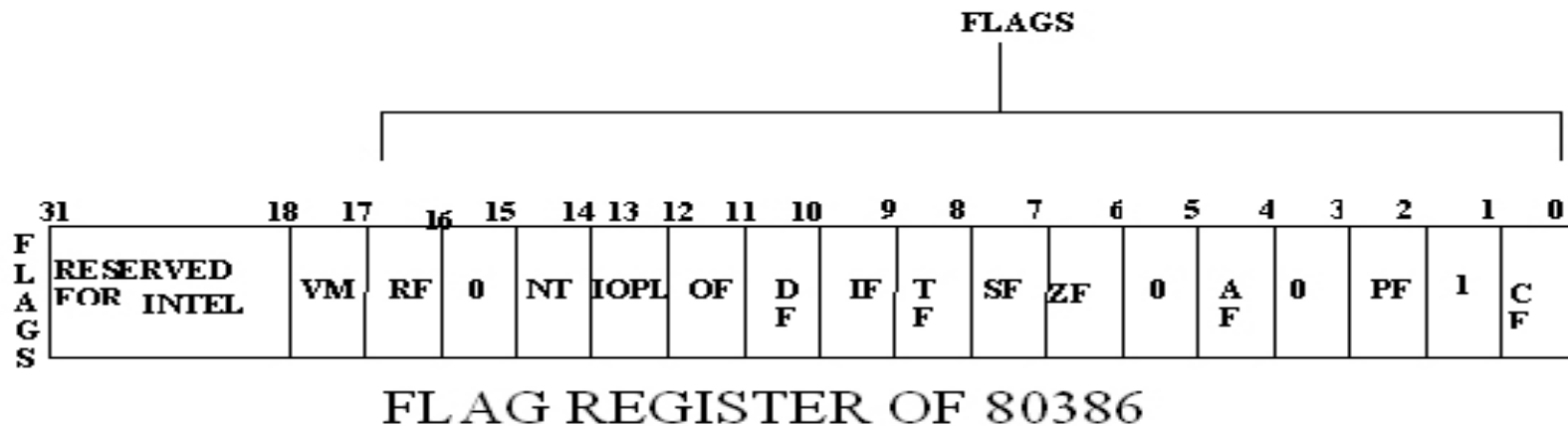
- Note:* 1. The address ranges illustrated in the page directory and page table represent the linear address ranges selected and not the contents of these tables.
2. The addresses (XXXXX) listed in the memory page are selected by the page table entry.

# The Task Register

- It is the key element in the protected mode task switching mechanism of the 80386dx microprocessor.
- This register holds a 16 bit index value called a selector.
- The initial selector must be loaded into TR under software control. This starts the initial task.
- After this is done, the selector is changed automatically whenever the processor executes an instruction that performs a task switch.
- The selector in TR is used to locate a descriptor in the GDT.
- When a selector is loaded into the TR, the corresponding Task State Segment (TSS) descriptor automatically gets read from the memory and loaded into the on-chip Task Descriptor Cache.
- This descriptor defines a block of memory called the Task state segment.
- It does this by providing the starting address (BASE) and the size-LIMIT of the segment.
- Every task has its own TSS.
- The TSS holds the information needed to initiate a task, such as initial values for the user accessible registers.

# Registers with changed functionality

- Register whose function changes when the 80386DX is switched to protected mode is the flag register.
- The flag register is now identified as the EFLAGS and expands to 32 bits in length.
- 5 additional bits have been included here.
- 2 I/O privilege level-IOPL, the nested task flag-NF, The resume flag-RF, the virtual 8086 mode flag-VM





### **VM - Virtual Mode Flag:**

- If this flag is set, the 80386 enters the virtual 8086 mode within the protection mode.
- This is to be set only when the 80386 is in protected mode.
- In this mode, if any privileged instruction is executed an exception 13 is generated.
- This bit can be set using IRET instruction or any task switch operation only in the protected mode.

### **RF- Resume Flag:**

- This flag is used with the debug register breakpoints.
- It is checked at the starting of every instruction cycle and if it is set, any debug fault is ignored during the instruction cycle.
- The RF is automatically reset after successful execution of every instruction, except for IRET and POPF instructions.
- Also, it is not automatically cleared after the successful execution of JMP, CALL and INT instruction causing a task switch.
- These instructions are used to set the RF to the value specified by the memory data available at the stack.

# Test and Debug Registers

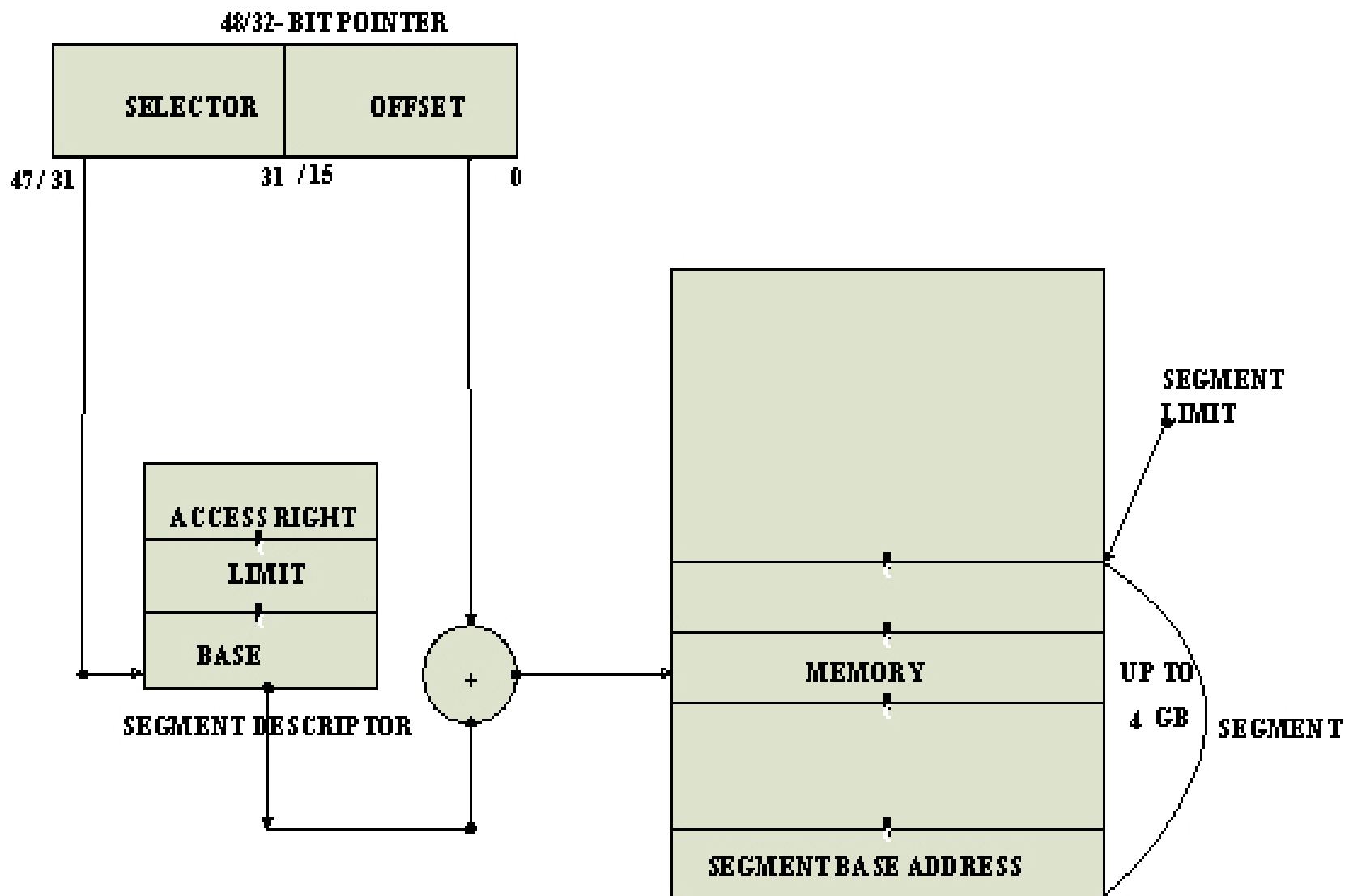
- Intel has provided a set of 8 debug registers for hardware debugging.
- Out of these eight registers DR<sub>0</sub> to DR<sub>7</sub>, two registers DR<sub>4</sub> and DR<sub>5</sub> are Intel reserved.
- The initial four registers DR<sub>0</sub> to DR<sub>3</sub> store four program controllable breakpoint addresses, while DR<sub>6</sub> and DR<sub>7</sub> respectively hold breakpoint status and breakpoint control information.
- Two more test register are provided by 80386 for page caching namely test control and test status register.

# *Multitasking and Protection*

- It contains on-chip hardware that permits multiple tasks to exist in a software system and allows them to be scheduled for execution in a time-shared manner.
- Safeguards can be built into the protected mode software system to deny the unauthorized access of a task's memory resource. This concept is called protection.
- The processor includes an on-chip hardware that implements a protection mechanism.
- The mechanism is designed to put restrictions on the access of local and system resources by a task and to isolate tasks from each other in a multitasking environment.

# *Physical Address Space and Virtual to physical Address Translation:*

- As a part of the translation process, the memory management unit determines whether or not the corresponding segment or page already resides in the memory, the operation is performed on the information.
- However if the segment or page is not present, it signals this condition as an error.
- Once this condition is identified, the memory manager software initiates loading of the segment or page from the external storage device to the physical memory. This operation is called SWAP.
- That is an old segment or page gets swapped out of the memory and then a new segment is swapped in the freed space in the physical memory.



Protected Mode Addressing Without Paging Unit

# Summary

- 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 pass it to the control section for deriving the necessary control signals. The barrel shifter increases the speed of all shift and rotate operations.
- The multiply / divide logic implements the bit-shift-rotate algorithms to complete the operations in minimum time.
- Even 32- bit multiplications can be executed within one microsecond by the multiply / divide logic.
- 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 relocability and sharing of code and data.
- Segmentation unit allows segments of size 4Gbytes at max.
- The 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. The virtual memory is also organizes in terms of segments and pages by the memory management unit.
- The Segmentation unit provides a 4 level protection mechanism for protecting and isolating the system code and data from those of the application program.
- 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. 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.
- The Bus control unit has a prioritizer to resolve the priority of the various bus requests. This controls the access of the bus. The address driver drives the bus enable and address signal A0 – A31. The pipeline and dynamic bus sizing unit handle the related control signals.
- The data buffers interface the internal data bus with the system bus.

Thank you