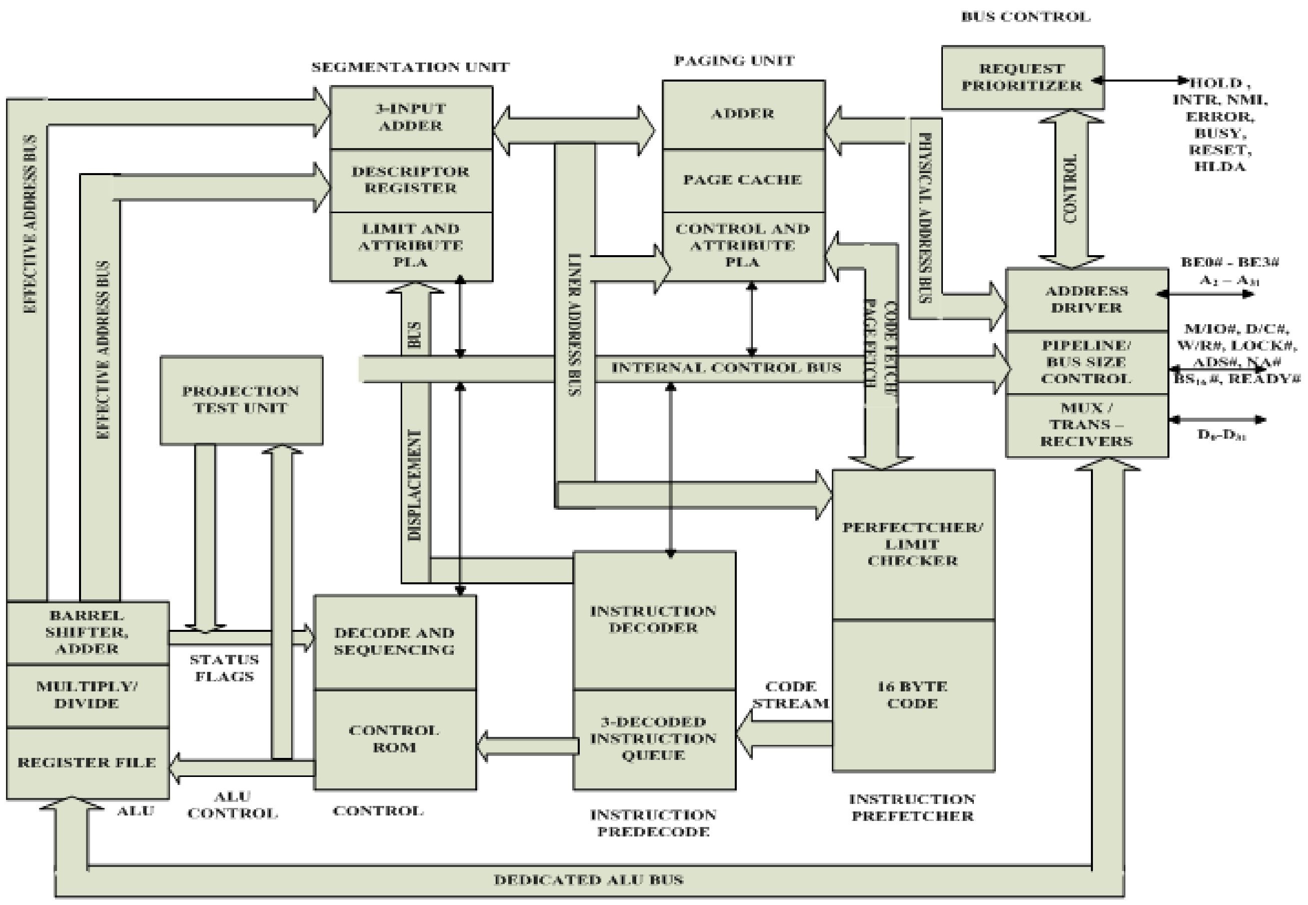
❖ Architecture of 80386

- The Internal Architecture of 80386 is divided into 3 sections.
    1. Central processing unit
    2. Memory management unit
    3. Bus interface unit
- Central processing unit is further divided into Execution unit and Instruction unit
- Execution unit has 8 General purpose and 8 Special purpose registers which are either used for handling data or calculating
  offset addresses.



80386 ARCHITECTURE

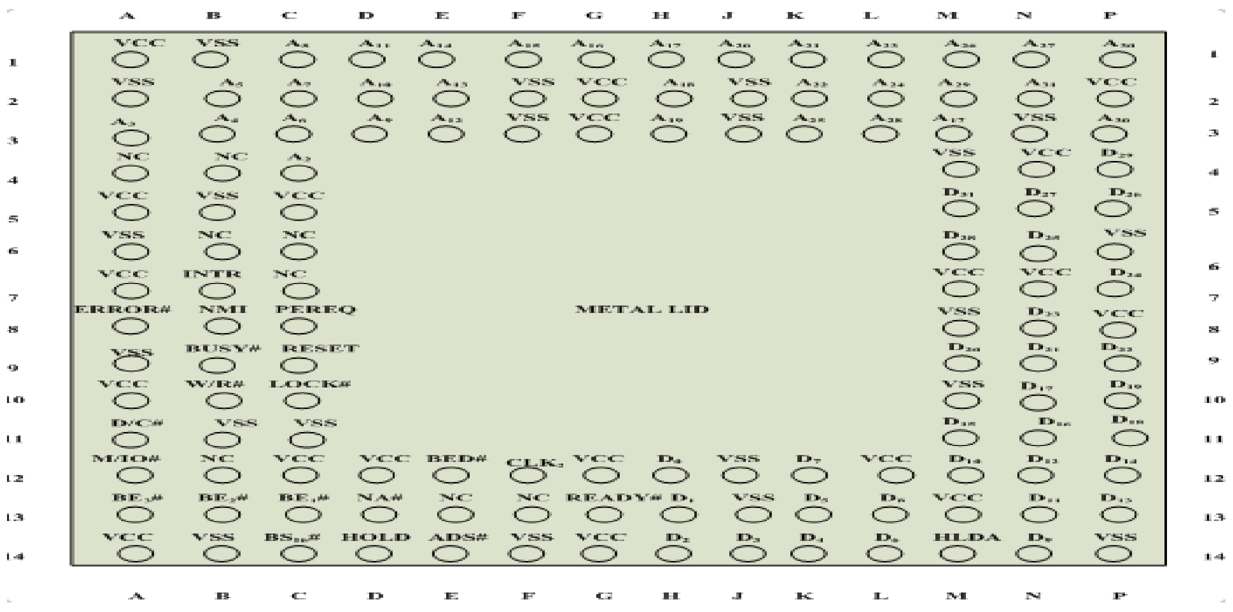
- The Instruction unit decodes the op-code bytes received from the 16-byte instruction code queue and arranges them in a 3- instruction decoded instruction queue.
- After decoding them pass it to the control section for deriving them necessary control signals. The barrel shifter increases the speed of 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 prioritize to resolve the priority of the

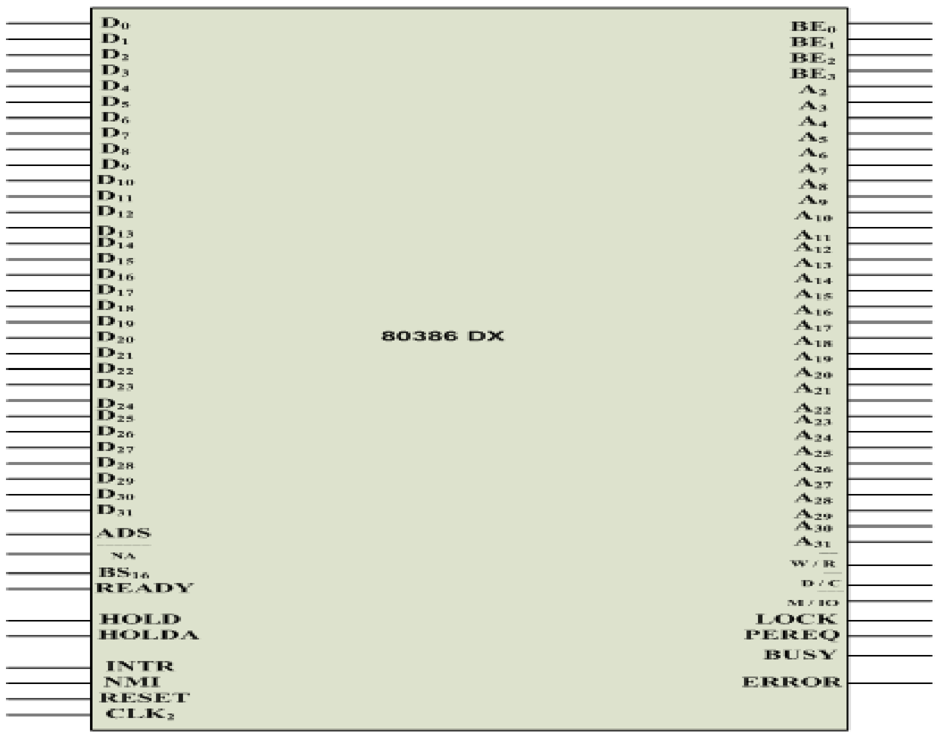
various bus requests. This controls the access of the bus. The address driver drives the bus enable and address signal A0 – A31. The pipeline and dynamic bus sizing unit handle the related control signals.

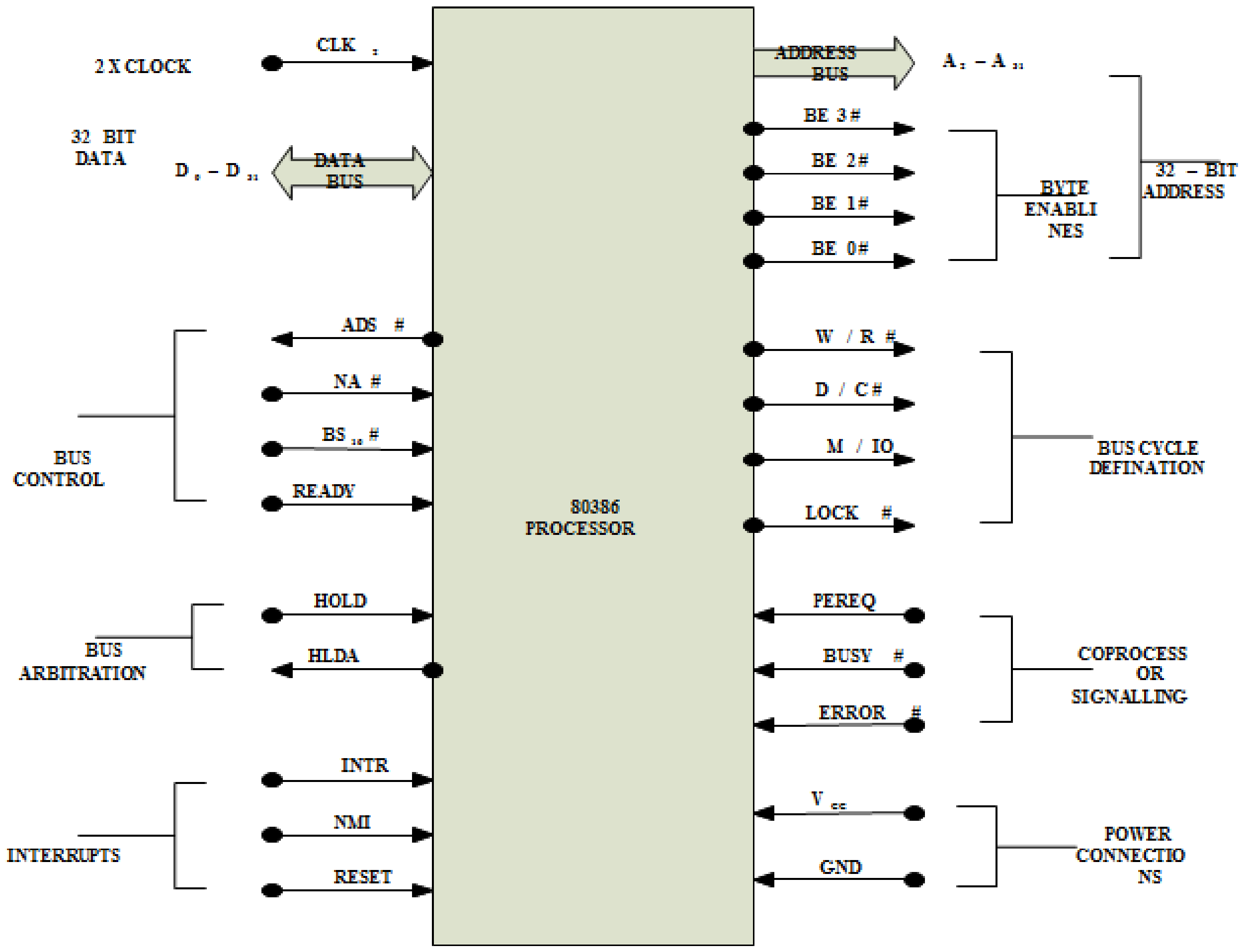
- The data buffers interface the internal data bus with the system bus.

❖ Signal Descriptions of 80386

- CLK2 :The input pin provides the basic system clock timing for the operation of 80386.
- D0 – D31:These 32 lines act as bidirectional data bus during different access cycles.
- A31 – A2: These are upper 30 bit of the 32- bit address bus.
- $\overline{\text{BE0 to BE3}}$: 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 $\overline{\text{BE0 to BE3}}$, may be used for enabling these 4 blanks. Using these 4 enable signal lines, the CPU may transfer 1 byte / 2 / 3 / 4 byte of data simultaneously.




80386 DX
D0 – D31
ADS
NA
BS16
READY
HOLD
HOLDA
INTR
NMI
RESET
CLK2
BE0 BE1 BE2 BE3
A2 – A31
W / R
D / C
M / IO
LOCK
PEREQ
BUSY
ERROR


2 X CLOCK
CLK2
32 BIT DATA
D0 – D31
DATA BUS
ADDRESS BUS
A2 – A31
BE 3#
BE 2#
BE 1#
BE 0#
BYTE ENABLINES
32 – BIT ADDRESS
BUS CONTROL
ADS #
NA #
BS16 #
READY
W / R #
D / C#
M / IO
LOCK #
BUS CYCLE DEFINATION
80386 PROCESSOR
BUS ARBITRATION
HOLD
HLDA
PEREQ
BUSY #
ERROR #
COPROCESSOR SIGNALLING
INTERRUPTS
INTR
NMI
RESET
Vcc
GND
POWER CONNECTIONS

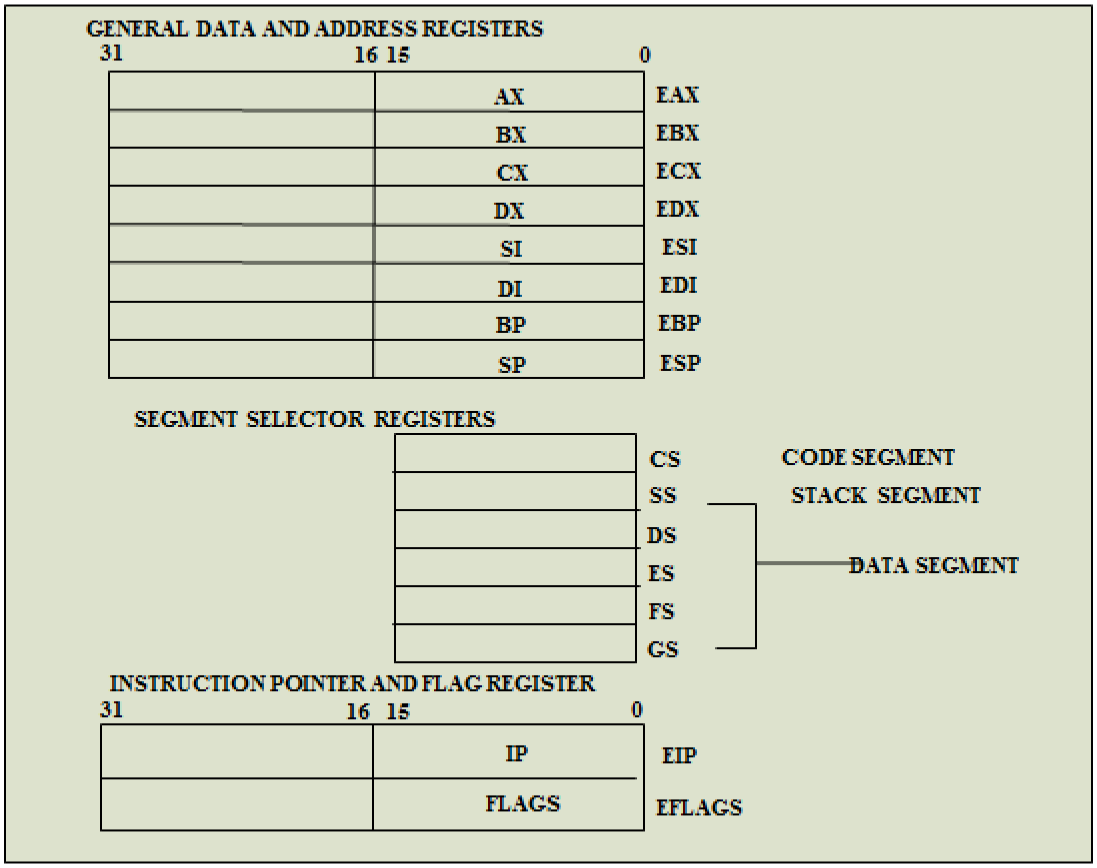
- **W/R:** The write / read output distinguishes the write and read cycles from one another.
- **D/C:** This data / control output pin distinguishes between a data transfer cycle from a machine control cycle like interrupt acknowledge.
- **M/IO:** This o/p pin diffs between the memory and I/O cycles.
- **LOCK:** The LOCK# output pin enables the CPU to prevent the other bus masters from gaining the control of the system bus.
- **NA:**The next address input pin, if activated, allows address pipelining, during 80386 bus cycles.
- **ADS:** The address status output pin indicates that the address bus and bus cycle definition pins(W/R#, D/C#, M/IO#, BE0# to BE3#) are carrying the respective valid signals. The 80383 does not have any ALE signals and so this signals may be used for latching the address to external latches.
- **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. The signal is used to insert WAIT states in a bus cycle and is useful for interfacing of slow devices with CPU.
- **VCC:** These are system power supply lines.
- **VSS:** These return lines for the power supply.
- **BS_{16}:** The bus size – 16 input pin allows the interfacing of 16 bit devices with the 32 bit wide 80386 data bus. Successive 16 bit bus cycles may be executed to read a 32 bit data from a peripheral.
- **HOLD**: The bus hold input pin enables the other bus masters to gain control of the system bus if it is asserted.
- **HLDA**: The bus hold acknowledge output indicates that a valid bus hold request has been received and the bus has been relinquished by the CPU.

- **BUSY:** The busy input signal indicates to the CPU that the coprocessor is busy with the allocated task.
- **ERROR:** The error input pin indicates to the CPU that the coprocessor has encountered an error while executing its instruction.
- **PEREQ**: The processor extension request output signal indicates to the CPU to fetch a data word for the coprocessor.
- **INTR**: This interrupt pin is a mask able interrupt, that can be masked using the IF of the flag register.
- **NMI:** A valid request signal at the non-mask able interrupt request input pin internally generates a non- mask able interrupt of type2.
- **RESET**: A high at this input pin suspends the current operation and restart the execution from the starting location.
- **N / C** : No connection pins are expected to be left open while connecting the 80386 in the circuit

❖ Register Organisation

- The 80386 has eight 32 - bit general purpose registers which may be used as either 8 bit or 16 bit registers.
- A 32 - bit register known as an extended register, is represented by the register name with prefix E.
- Example : A 32 bit register corresponding to AX is EAX, similarly BX is EBX etc.
- The 16 bit registers BP, SP, SI and DI in 8086 are now available with their extended size of 32 bit and are names as EBP,ESP,ESI and EDI.

- AX represents the lower 16 bit of the 32 bit register EAX.
- BP, SP, SI, DI represents the lower 16 bit of their 32 bit counterparts, and can be used as independent 16 bit registers

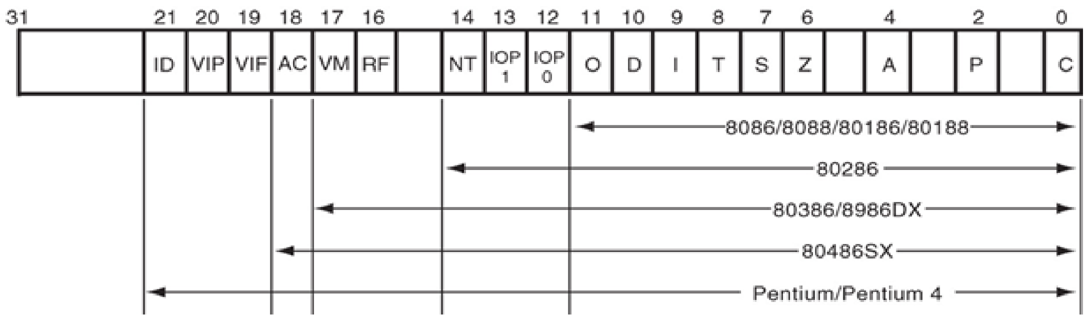


- The six segment registers available in 80386 are CS, SS, DS, ES, FS and GS.
- The CS and SS are the code and the stack segment registers respectively, while DS, ES, FS, GS are 4 data segment registers.
- A 16 bit instruction pointer IP is available along with 32 bit counterpart EIP.

❖ Flag Register of 80386:

- **Flag Register of 80386**: The Flag register of 80386 is a 32 bit register. Out of the 32 bits, Intel has reserved bits D18 to D31, D5 and D3, while D1 is always set at 1.Two extra new flags are added to the 80286 flag to derive the flag register of 80386. They are VM and RF flags.

Figure 2–2 The EFLAG and FLAG register counts for the entire 8086 and Pentium microprocessor family.



- Flags never change for any data transfer or program control operation.
- Some of the flags are also used to control features found in the microprocessor.

- <u>**Flag bits, with a brief description of function.**</u>
- **C (carry)** holds the carry after addition or borrow after subtraction.
  -also indicates error conditions
- **P (parity)** is the count of ones in a number expressed as even or odd. Logic 0 for odd parity; logic 1 for even parity.
  -if a number contains three binary one bits, it has odd parity
  -if a number contains no one bits, it has even parity
- **A (auxiliary carry)** holds the carry after addition or the borrow after subtraction between bit positions 3 and 4 of the result.

- **Z (zero)** shows that the result of an arithmetic or logic operation is zero.
- **S (sign)** flag holds the arithmetic sign of the result after an arithmetic or logic instruction executes.
- **T (trap)** The trap flag enables trapping through an on-chip debugging feature.
- **I (interrupt)** controls operation of the INTR (interrupt request) input pin.
- **D (direction)** selects increment or decrement mode for the DI and/or SI registers.
- **O (overflow)** occurs when signed numbers are added or subtracted.

  -an overflow indicates the result has exceeded the capacity of the machine
- **IOPL** used in protected mode operation to select the privilege level for I/O devices.
- **NT (nested task)** flag indicates the current task is nested within another task in protected mode operation.
- **RF (resume)** used with debugging to control resumption of execution after the next instruction.
- **VM (virtual mode)** flag bit selects virtual mode operation in a protected mode system
- **AC, (alignment check)** flag bit activates if a word or double word is addressed on a non- word or non-double word boundary.
- **VIF** is a copy of the interrupt flag bit available to the Pentium 4
- **VIP (virtual)** provides information about a virtual mode interrupt for (interrupt pending) Pentium.

  -used in multitasking environments to provide virtual interrupt flags

- **ID (identification)** flag indicates that the Pentium microprocessors support the CPUID instruction.
  - CPUID instruction provides the system with information about the Pentium microprocessor
- **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 instruction are used to set the RF to the value specified by the memory data available at the stack.
- **Segment Descriptor Registers:** This registers are not available for programmers, rather they are internally used to store the descriptor information, like attributes, limit and base addresses of segments.

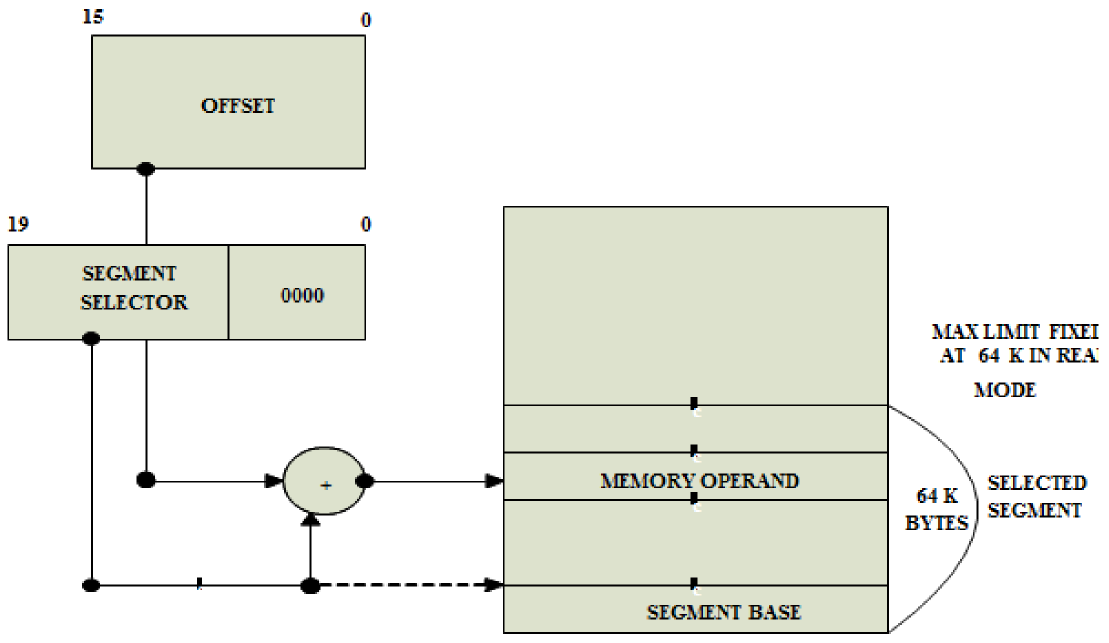
- The six segment registers have corresponding six 73 bit descriptor registers. Each of them contains 32 bit base address, 32 bit base limit and 9 bit attributes. These are automatically loaded when the corresponding segments are loaded with selectors.
- **Control Registers:** The 80386 has three 32 bit control registers CR), CR2 and CR3 to hold global machine status independent of the executed task. Load and store instructions are available to access these registers.
- **System Address Registers:** Four special registers are defined to refer to the descriptor tables supported by 80386.
- The 80386 supports four types of descriptor table, viz. global descriptor table (GDT), interrupt descriptor table (IDT), local descriptor table (LDT) and task state segment descriptor (TSS).
- **Debug and Test Registers:** Intel has provide a set of 8 debug registers for hardware debugging. Out of these eight registers DR0 to DR7, two registers DR4 and DR5 are Intel reserved.
- The initial four registers DR0 to DR3 store four program controllable breakpoint addresses, while DR6 and DR7 respectively hold breakpoint status and breakpoint control information.
- Two more test register are provided by 80386 for page cacheing namely test control and test status register.

❖ ADDRESSING MODES

- **ADDRESSING MODES:** The 80386 supports overall eleven addressing modes to facilitate efficient execution of higher level language programs.
- In case of all those modes, the 80386 can now have 32-bit immediate or 32- bit register operands or displacements.
- The 80386 has a family of scaled modes. In case of scaled modes, any of the index register values can be multiplied by a valid scale factor to obtain the displacement.
- The valid scale factor are 1, 2, 4 and 8.
- The different scaled modes are as follows.
- **Scaled Indexed Mode**: Contents of the an index register are multiplied by a scale factor that may be added further to get the operand offset.
- **Based Scaled Indexed Mode**: Contents of the an index register are multiplied by a scale factor and then added to base register to obtain the offset.
- **Based Scaled Indexed Mode with Displacement**: The Contents of the an index register are multiplied by a scaling factor and the result is added to a base register and a displacement to get the offset of an operand.

❖ Real Address Mode of 80386

- After reset, the 80386 starts from memory location FFFFFFF0H under the real address mode. In the real mode, 80386 works as a fast 8086 with 32-bit registers and data types.
- In real mode, the default operand size is 16 bit but 32- bit operands and addressing modes may be used with the help of override prefixes.
- The segment size in real mode is 64k, hence the 32-bit effective addressing must be less than 0000FFFFFH. The real mode initializes the 80386 and prepares it for protected mode.



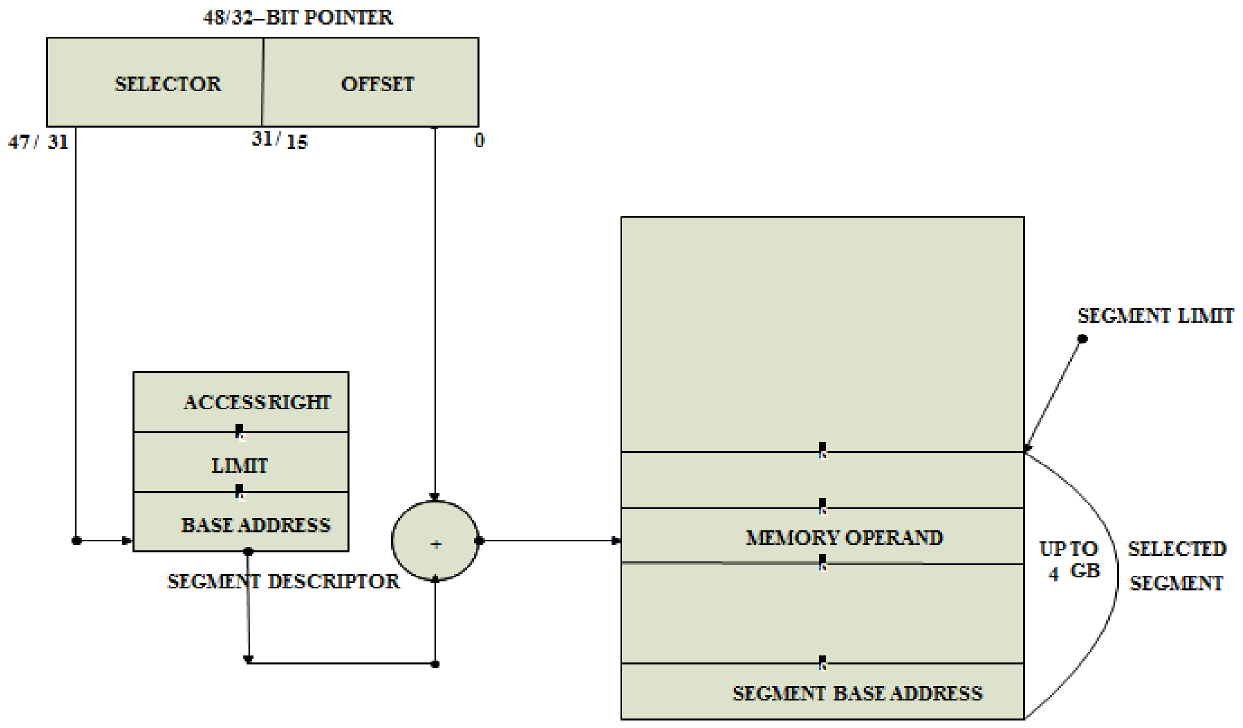
Physical Address Formation In Real Mode Of 80386

- **Memory Addressing in Real Mode:** In the real mode, the 80386 can address at the most 1Mbytes of physical memory using address lines A0-A19.
- Paging unit is disabled in real addressing mode, and hence the real addresses are the same as the physical addresses.
- To form a physical memory address, appropriate segment registers contents (16-bits) are shifted left by four positions and then added to the 16-bit offset address formed using one of the addressing modes, in the same way as in the 80386 real address mode.
- The segment in 80386 real mode can be read, write or executed, i.e. no protection is available
- Any fetch or access past the end of the segment limit generate exception 13 in real address mode.
- The segments in 80386 real mode may be overlapped or non-overlapped.
- The interrupt vector table of 80386 has been allocated 1Kbyte space starting from 00000H to 003FFH.

❖ Protected Mode of 80386

- All the capabilities of 80386 are available for utilization in its protected mode of operation.
- The 80386 in protected mode support all the software written for 80286 and 8086 to be executed under the control of memory management and protection abilities of 80386.
- The protected mode allows the use of additional instruction, addressing modes and capabilities of 80386.

•**ADDRESSING IN PROTECTED MODE**: In this mode, the contents of segment registers are used as selectors to address descriptors which contain the segment limit, base address and access rights byte of the segment.

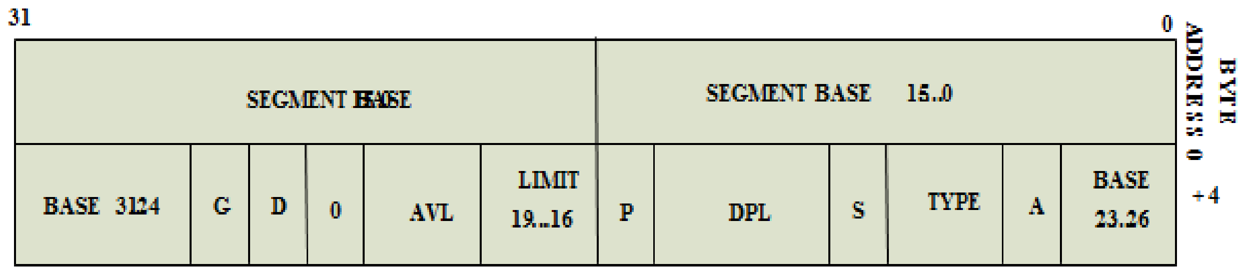


Protected Mode Addressing Without Paging Unit

- The effective address (offset) is added with segment base address to calculate linear address. This linear address is further used as physical address, if the paging unit is disabled, otherwise the paging unit converts the linear address into physical address.
- The paging unit is a memory management unit enabled only in protected mode. The paging mechanism allows handling of large segments of memory in terms of pages of 4Kbyte size.
- The paging unit operates under the control of segmentation unit. The paging unit if enabled converts linear addresses into physical address, in protected mode.

❖ Segmentation

- DESCRIPTOR TABLES: These descriptor tables and registers are manipulated by the operating system to ensure the correct operation of the processor, and hence the correct execution of the program.
- Three types of the 80386 descriptor tables are listed as follows:
  GLOBAL DESCRIPTOR TABLE (GDT)
  LOCAL DESCRIPTOR TABLE (LDT)
  INTERRUPT DESCRIPTOR TABLE (IDT)
- **DESCRIPTORS:** The 80386 descriptors have a 20-bit segment limit and 32-bit segment address. The descriptor of 80386 are 8-byte quantities access right or attribute bits along with the base and limit of the segments.
- **Descriptor Attribute Bits:** The A (accessed) attributed bit indicates whether the segment has been accessed by the CPU or not.
- The TYPE field decides the descriptor type and hence the segment type.
- The S bit decides whether it is a system descriptor (S=0) or code/data segment descriptor (S=1).



- The DPL field specifies the descriptor privilege level.
- The D bit specifies the code segment operation size. If D=1, the segment is a 32-bit operand segment, else, it is a 16-bit operand segment.
- The P bit (present) signifies whether the segment is present in the physical memory or not. If P=1, the segment is present in the physical memory.
- The G (granularity) bit indicates whether the segment is page addressable. The zero bit must remain zero for compatibility with future process.
- The AVL (available) field specifies whether the descriptor is for user or for operating system.
- The 80386 has five types of descriptors listed as follows:
  1.Code or Data Segment Descriptors.
  2.System Descriptors.
  3.Local descriptors.
  4.TSS (Task State Segment) Descriptors.
  5.GATE Descriptors.
- The 80386 provides a four level protection mechanism exactly in the same way as the 80286 does.

❖ Paging

- **PAGING OPERATION**: Paging is one of the memory management techniques used for virtual memory multitasking operating system.
- The segmentation scheme may divide the physical memory into a variable size segments but the paging divides the memory into a fixed size pages.
- The segments are supposed to be the logical segments of the program, but the pages do not have any logical relation with the program.
- The pages are just fixed size portions of the program module or data.
- The advantage of paging scheme is that the complete segment of a task need not be in the physical memory at any time.
- Only a few pages of the segments, which are required currently for the execution need to be available in the physical memory. Thus the memory requirement of the task is substantially reduced, relinquishing the available memory for other tasks.
- Whenever the other pages of task are required for execution, they may be fetched from the secondary storage.
- The previous page which are executed, need not be available in the memory, and hence the space occupied by them may be relinquished for other tasks.
- Thus paging mechanism provides an effective technique to manage the physical memory for multitasking systems.

- **Paging Unit:** The paging unit of 80386 uses a two level table mechanism to convert a linear address provided by segmentation unit into physical addresses.
- The paging unit converts the complete map of a task into pages, each of size 4K. The task is further handled in terms of its page, rather than segments.
- The paging unit handles every task in terms of three components namely page directory, page tables and page itself.
- **Paging Descriptor Base Register:** The control register CR2 is used to store the 32-bit linear address at which the previous page fault was detected.
- The CR3 is used as page directory physical base address register, to store the physical starting address of the page directory.
- The lower 12 bit of the CR3 are always zero to ensure the page size aligned directory. A move operation to CR3 automatically loads the page table entry caches and a task switch operation, to load CR0 suitably.
- **Page Directory :** This is at the most 4Kbytes in size. Each directory entry is of 4 bytes, thus a total of 1024 entries are allowed in a directory.
- The upper 10 bits of the linear address are used as an index to the corresponding page directory entry. The page directory entries point to page tables.
- **Page Tables:** Each page table is of 4Kbytes in size and many contain a maximum of 1024 entries. The page table entries contain the starting address of the page and the statistical

information about the page.

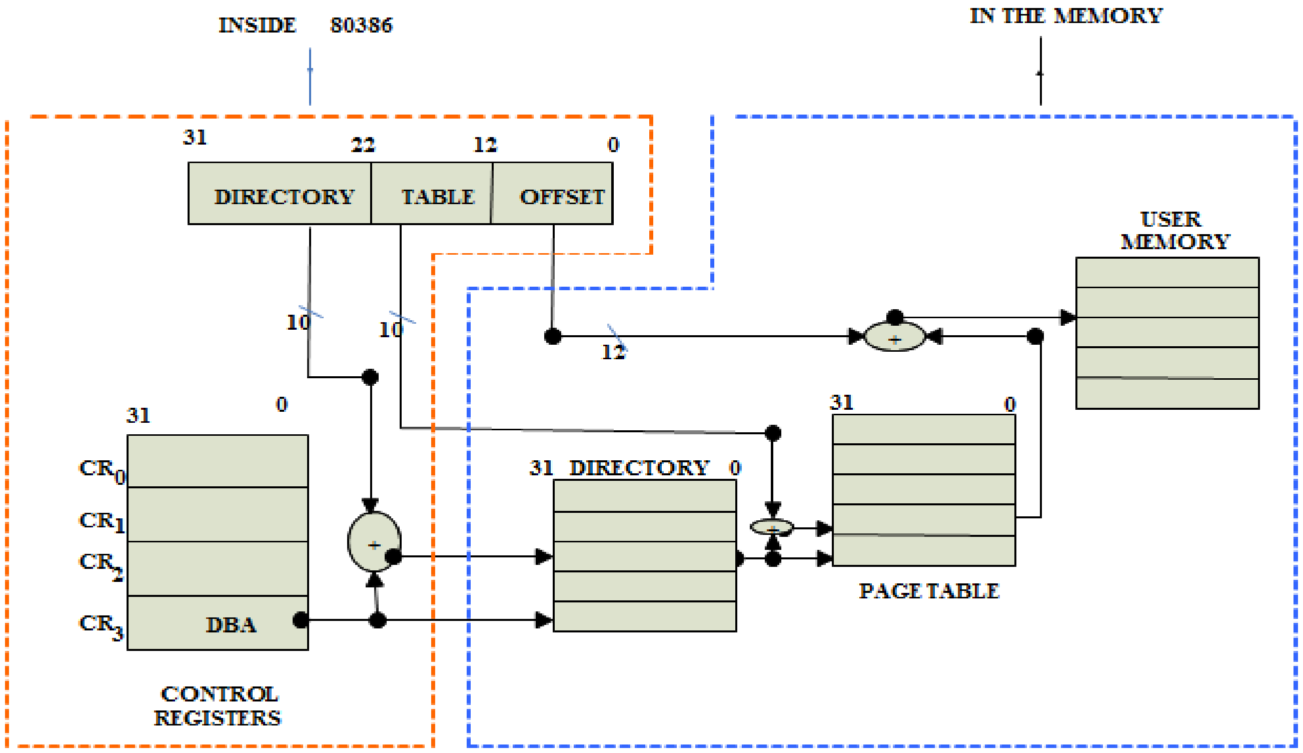
PAGE TABLE ENTRY

$U/\bar{S}$	$R/\bar{W}$	PERMITTED FOR LEVEL 3	PERMITTED FOR LEVEL 2 ,1OR 0
0	0	NONE	READ / WRITE
0	1	NONE	READ / WRITE
1	0	READ ONLY	READ / WRITE
1	1	READ - WRITE	READ / WRITE

PAGE DIRECTORY ENTRY

PAGE TABLE ADDRESS 31...12	OS RESERVED	0	0	D	A	0	0	$U/\bar{S}$	$R/\bar{W}$	P

- The upper 20 bit page frame address is combined with the lower 12 bit of the linear address. The address bits A12- A21 are used to select the 1024 page table entries. The page table can be shared between the tasks.
- The P bit of the above entries indicate, if the entry can be used in address translation.
- If P=1, the entry can be used in address translation, otherwise it cannot be used.
- The P bit of the currently executed page is always high.
- The accessed bit A is set by 80386 before any access to the page. If A=1, the page is accessed, else unaccessed.

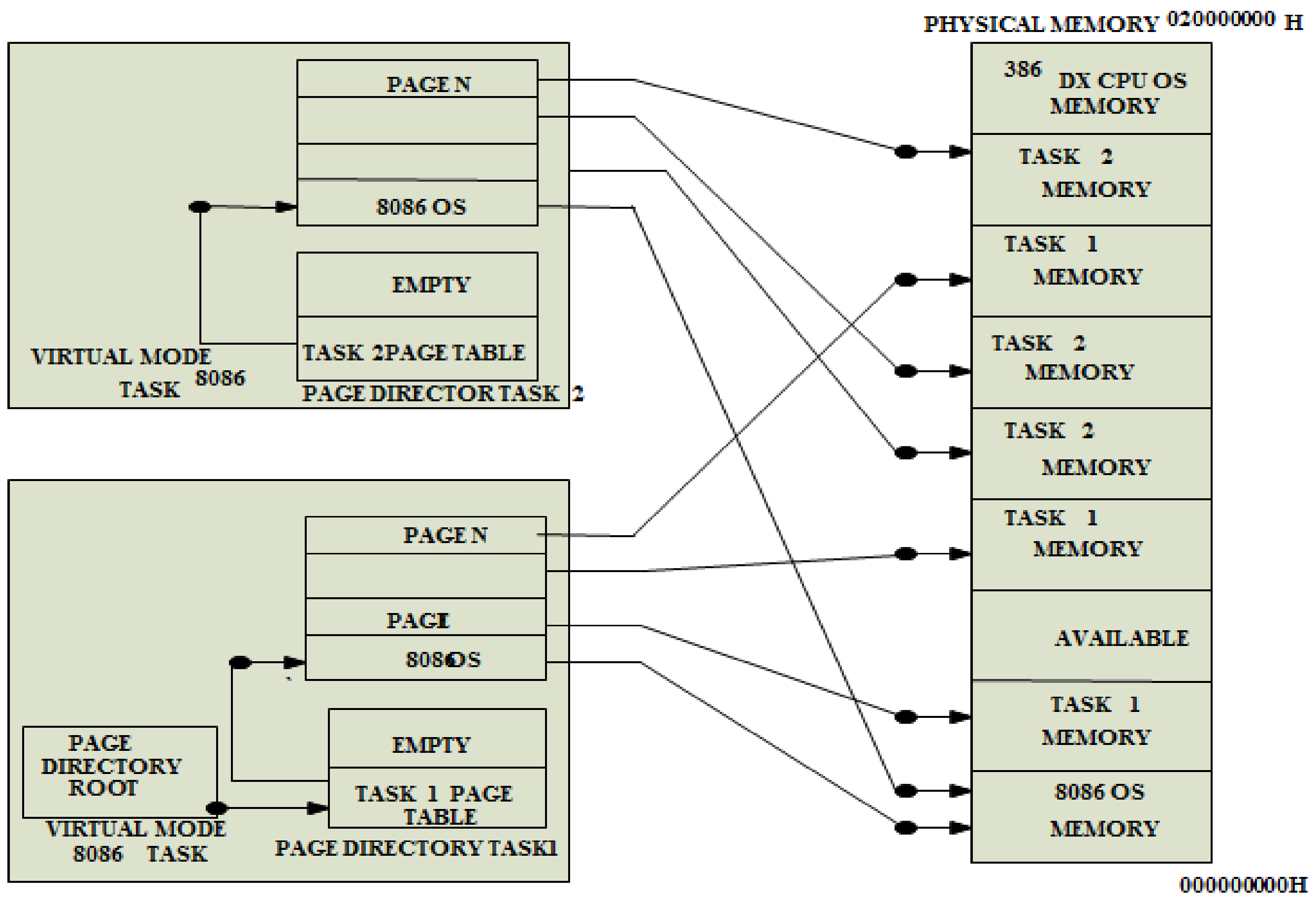


DBA Physical directory base address

- The D bit (Dirty bit) is set before a write operation to the page is carried out. The D-bit is undefined for page director entries.
- The OS reserved bits are defined by the operating system software.
- The User / Supervisor (U/S) bit and read/write bit are used to provide protection. These bits are decoded to provide protection under the 4 level protection model.
- The level 0 is supposed to have the highest privilege, while the level 3 is supposed to have the least privilege.
- This protection provide by the paging unit is transparent to the segmentation unit.

❖ Virtual 8086 Mode

- In its protected mode of operation, 80386DX provides a virtual 8086 operating environment to execute the 8086 programs.
- The real mode can also used to execute the 8086 programs along with the capabilities of 80386, like protection and a few additional instructions.
- Once the 80386 enters the protected mode from the real mode, it cannot return back to the real mode without a reset operation.
- Thus, the virtual 8086 mode of operation of 80386, offers an advantage of executing 8086 programs while in protected mode.
- The address forming mechanism in virtual 8086 mode is exactly identical with that of 8086 real mode.
- In virtual mode, 8086 can address 1Mbytes of physical memory that may be anywhere in the 4Gbytes address space of the protected mode of 80386.
- Like 80386 real mode, the addresses in virtual 8086 mode lie within 1Mbytes of memory.
- In virtual mode, the paging mechanism and protection capabilities are available at the service of the programmers.
- The 80386 supports multiprogramming, hence more than one programmer may be use the CPU at a time.



Memory Management In Virtual 8086 Mode

- Paging unit may not be necessarily enable in virtual mode, but may be needed to run the 8086 programs which require more than 1Mbyts of memory for memory managementfunction.
- In virtual mode, the paging unit allows only 256 pages, each of 4Kbytes size.
- Each of the pages may be located anywhere in the maximum 4Gbytes physical memory. The virtual mode allows the multiprogramming of 8086 applications.
- The virtual 8086 mode executes all the programs at privilege level 3.Any of the other programmers may deny access to the virtual mode programs or data.

- However, the real mode programs are executed at the highest privilege level, i.e. level 0.
- The virtual mode may be entered using an IRET instruction at CPL=0 or a task switch at any CPL, executing any task whose TSS is having a flag image with VM flag set to 1.
- The IRET instruction may be used to set the VM flag and consequently enter the virtual mode.
- The PUSHF and POPF instructions are unable to read or set the VM bit, as they do not access it.
- Even in the virtual mode, all the interrupts and exceptions are handled by the protected mode interrupthandler.
- To return to the protected mode from the virtual mode, any interrupt or execution may be used.
- As a part of interrupt service routine, the VM bit may be reset to zero to pull back the 80386 into protected mode.

❖ Features of 80386

- This 80386 is a 32bit processor that supports, 8bit/32bit data operands.
- The 80386 instruction set is upward compatible with all its predecessors.
- The 80386 can run 8086 applications under protected mode in its virtual 8086 mode of operation.
- With the 32 bit address bus, the 80386 can address upto 4Gbytes of physical memory. The physical memory is organized in terms of segments of 4Gbytes at maximum.

- The 80386 CPU supports 16K number of segments and thus the total virtual space of 4Gbytes * 16K = 64 Terrabytes.
- The memory management section of 80386 supports the virtual memory, paging and four levels of protection, maintaining full compatibility with80286.
- The 80386 offers a set of 8 debug registers DR_0-DR_7 for hardware debugging and control. The 80386 has on-chip address translation cache.
- The concept of paging is introduced in 80386 that enables it to organise the available physical memory in terms of pages of size 4Kbytes each, under the segmented memory.
- The 80386 can be supported by 80387 for mathematical data processing.

Chapter 2 -Introduction to Pentium processor

❖Pentium Processor

- Pentium Processor
  - 32-bit Microprocessor
    - 32-bit addressing
    - 64-bit Data Bus
  - Superscalar architecture
    - Two pipelined integer units
    - Capable of under one clock per instruction
    - Pipelined Floating Point Unit
  - Separate Code and Data Caches
    - 8K Code, 8K Write Back Data
    - 2-way 32-byte line size
    - MESI cache consistency protocol
  - Advance Design Features
    - Branch Prediction
  - 237-pin PGA

❖Internal Architecture of the Pentium Processors

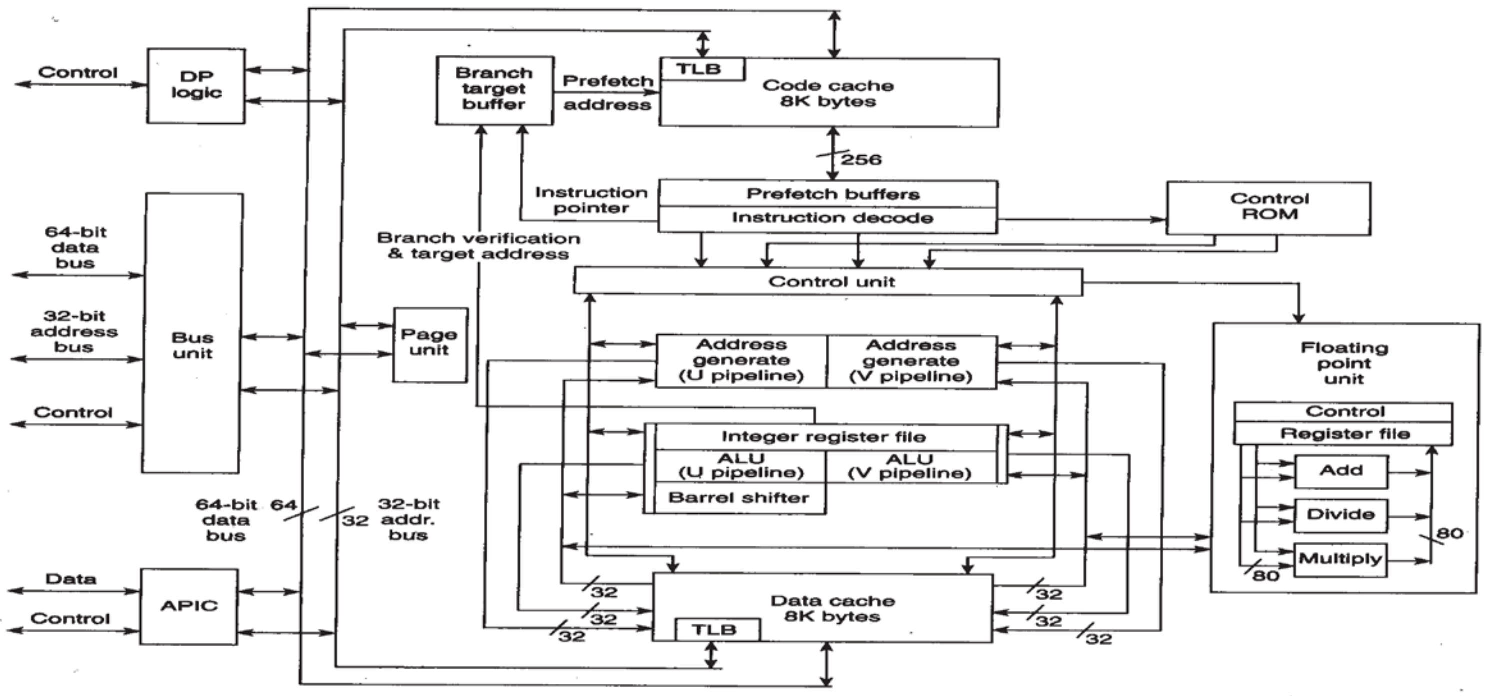


Figure 16.2 Internal architecture of the Pentium^R processors. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1994)

❖ Pentium Processor

❖ Pipeline and Instruction Flow

-5 stage pipeline

PF : prefetch

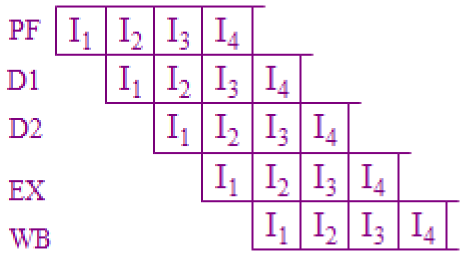
D1 : Instruction decode

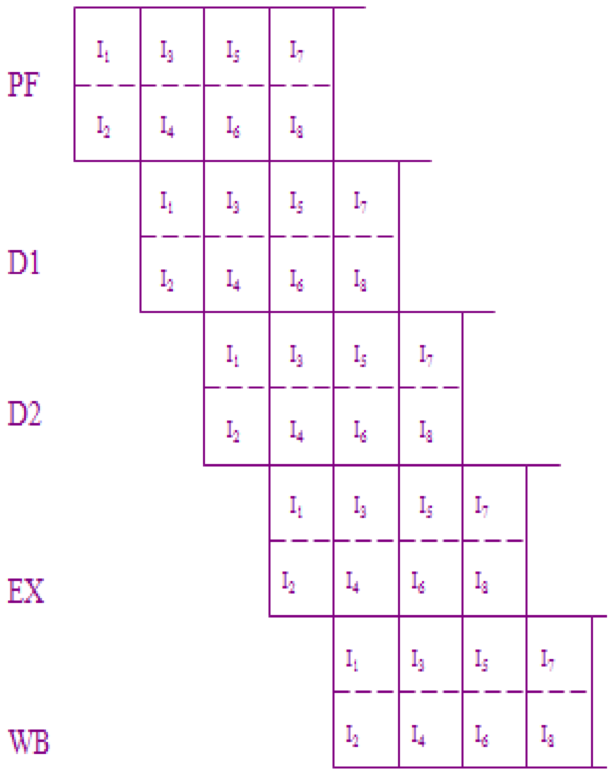
D2 : Address Generation

EX : Execute -ALU and Cache Access

WB : Write Back

Intel 486 Pentium





– “U”, “V” pipes - “pairing”
  - U : any instruction
  - V : ‘simple instructions” as defined in the ‘Pairing” rules

PF : instructions on chip cache or memory -> prefetch buffers prefetch buffers - two independent pairs of line size(32 bytes)

D1 : two parallel decoders

D2 : address generation for operand fetch

EX : ALU operations and data cache access

WB : modify processor state ; complete execution

❖Branch Prediction

- Branch Prediction

  –Branch Target Buffer

  –The processor accesses the BTB with the address of the instruction in the D1 stage

  - example:

  inner_loop :

mov	byte ptr flag[edx], al	PF	D1	D2	EX	WB		
add	edx, ecx		PF	D1	D2	EX	WB	
cmp	edx, FALSE			PF	D1	D2	EX	WB
jle	inner_loop				PF			

  – 486 : 6 clocks

  Pentium : 2 clocks with branch prediction

- EFLAGS

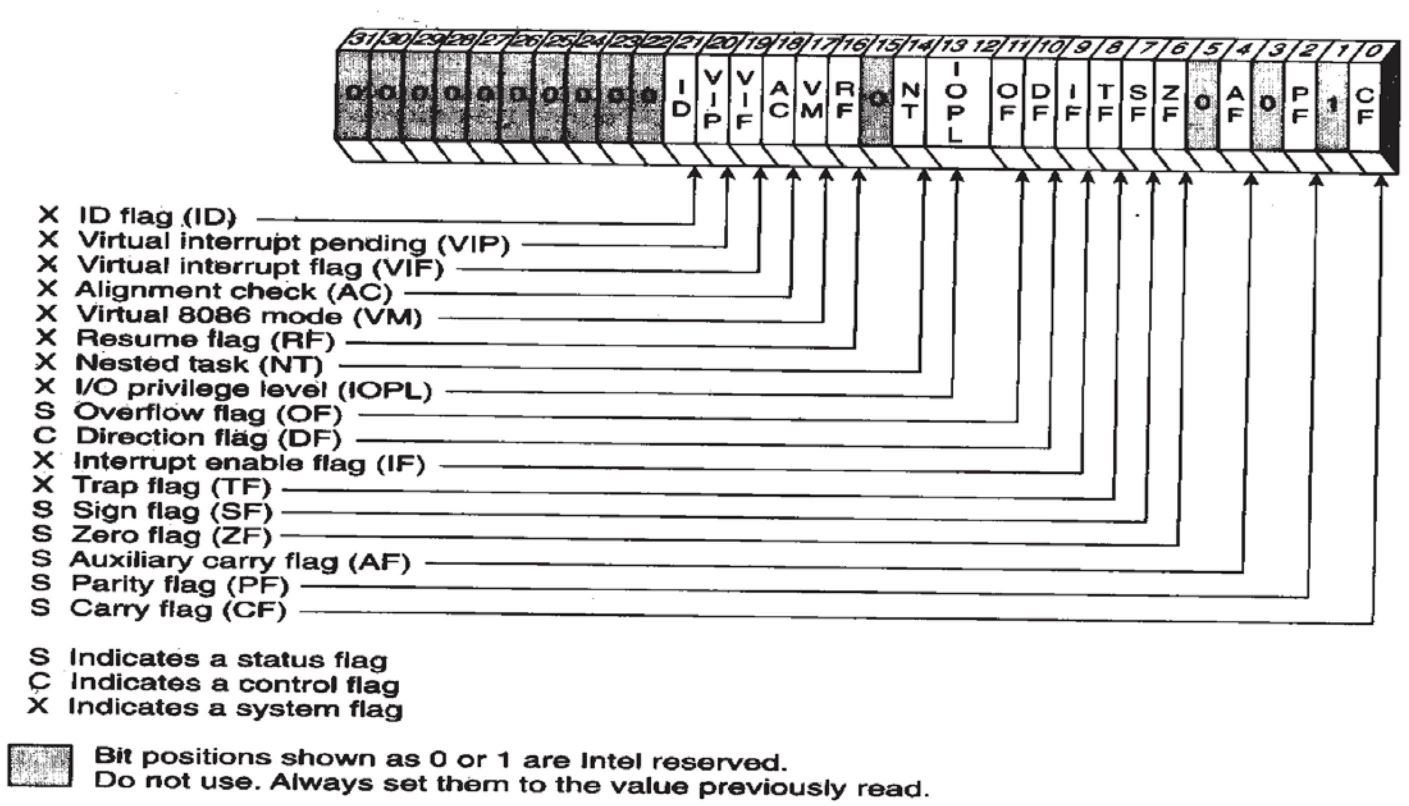
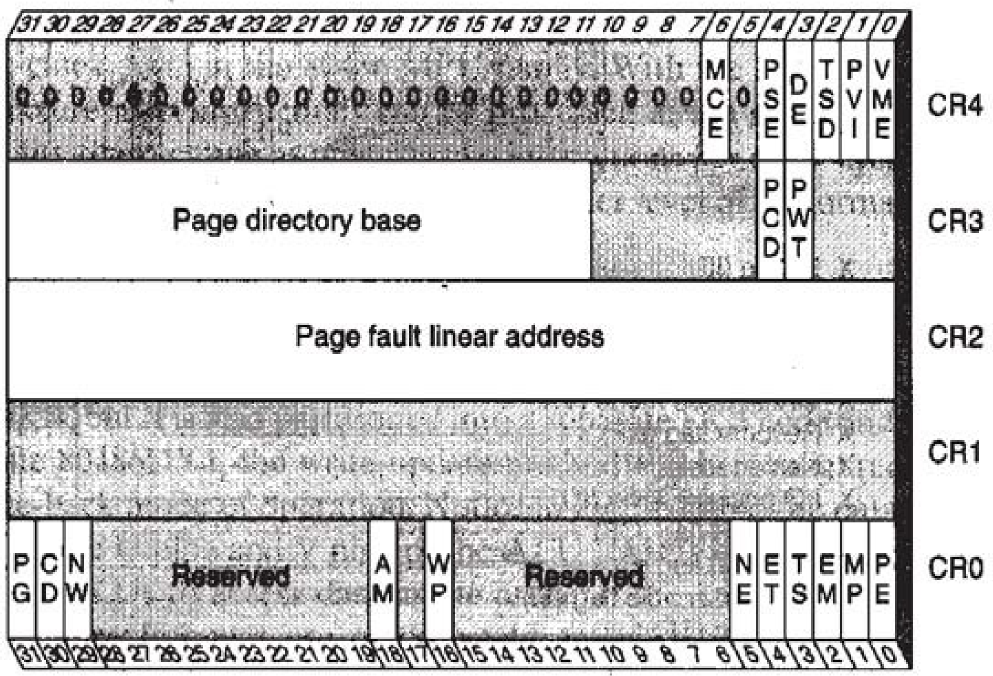
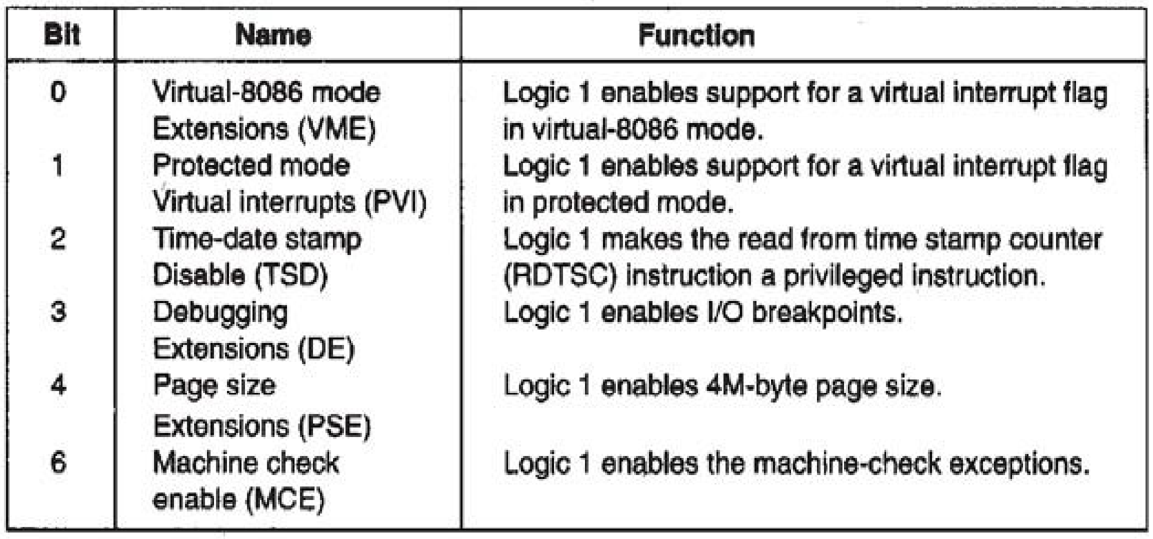

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0 0 0 ID VIP VIF AC VM RF 0 NT IOPL OF DF IF TF SF ZF 0 AF 0 PF 1 CF
X ID flag (ID)
X Virtual interrupt pending (VIP)
X Virtual interrupt flag (VIF)
X Alignment check (AC)
X Virtual 8086 mode (VM)
X Resume flag (RF)
X Nested task (NT)
X I/O privilege level (IOPL)
S Overflow flag (OF)
C Direction flag (DF)
X Interrupt enable flag (IF)
X Trap flag (TF)
S Sign flag (SF)
S Zero flag (ZF)
S Auxiliary carry flag (AF)
S Parity flag (PF)
S Carry flag (CF)
S Indicates a status flag
C Indicates a control flag
X Indicates a system flag
Bit positions shown as 0 or 1 are Intel reserved.
Do not use. Always set them to the value previously read.

Figure 16.3 EFLAGS registers of the Pentium^R processors. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1995)

- Control Registers of the Pentium Processor



(a)

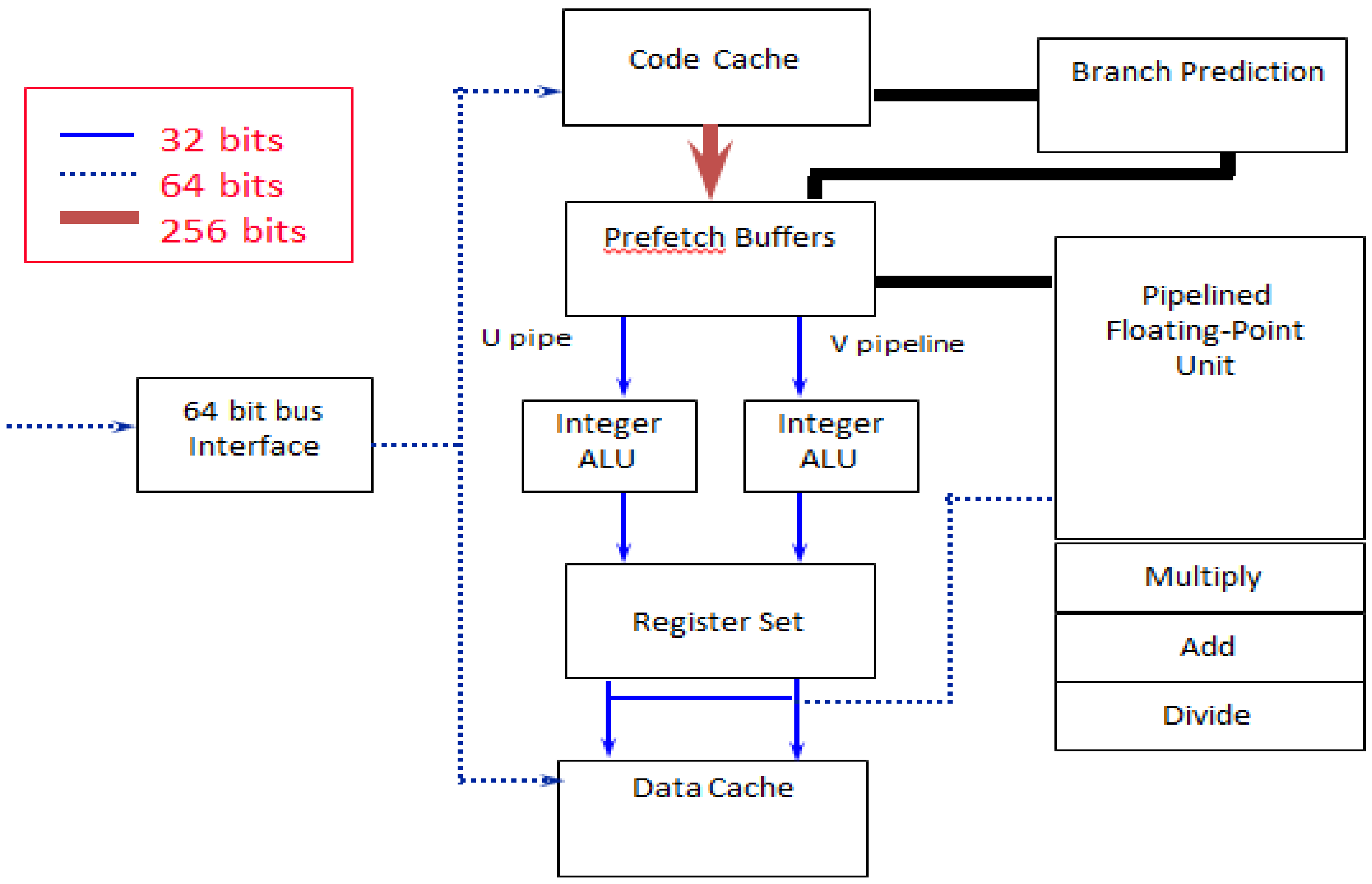


Bit	Name	Function
0	Virtual-8086 mode Extensions (VME)	Logic 1 enables support for a virtual interrupt flag in virtual-8086 mode.
1	Protected mode Virtual interrupts (PVI)	Logic 1 enables support for a virtual interrupt flag in protected mode.
2	Time-date stamp Disable (TSD)	Logic 1 makes the read from time stamp counter (RDTSC) instruction a privileged instruction.
3	Debugging Extensions (DE)	Logic 1 enables I/O breakpoints.
4	Page size Extensions (PSE)	Logic 1 enables 4M-byte page size.
6	Machine check enable (MCE)	Logic 1 enables the machine-check exceptions.

(b)

Figure 16.4 (a) Control registers of the Pentium^R processors. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1995) (b) Function of the CR_4 control bits.

❖ Pentium Processor Architecture



- The Pentium processors have a data bus of 64 bits.
  - This is a 32 bit CPU due to having 32 bits registers.
  - A standard Single Transfer Cycle can read or write up to 64 bits at a time (8 bytes)
- Burst read and burst write-back cycles are supported by the Pentium processors.
  - Burst Mode cycles are used for Cache operations and transfer 32 bytes in 4 clocks (4 * 8 bytes = 4 * 64 bits).
- 32 bytes is the size of the Pentium Cache line.
  - For the Pentium, all cache operations are burst cycles.

- Pentium processors include separate Code and Data Caches which can be enabled or disabled by software or hardware.
  - Each cache is 8-Kbytes in size, with a 32-byte line size and is 2-way set associative (4K/way).
  - The Data Cache is configurable to be write-back or write-through on a line-by-line basis and follows MESI protocol.
  - The Instruction Cache is an inherently write- protected cache (read-only)
- Instructions are Fetched from the code cache or from the external bus.
- The decode unit Decodes the prefetched instructions so the Pentium processor can execute the instruction.
  - Branch prediction is implemented with 2 Prefetch Buffers and a Branch Target Buffer so the needed code is almost always prefetched before it is needed for execution.
- Instructions are executed in 1 of 2 pipelines ("u" & "v" pipes) which share access to a single set of registers.
  - No additional instructions can begin execution until both execution units complete their operations.
- Pentium processors have two instruction pipelines.

– The u-pipe can Execute all integer and floating point instructions.

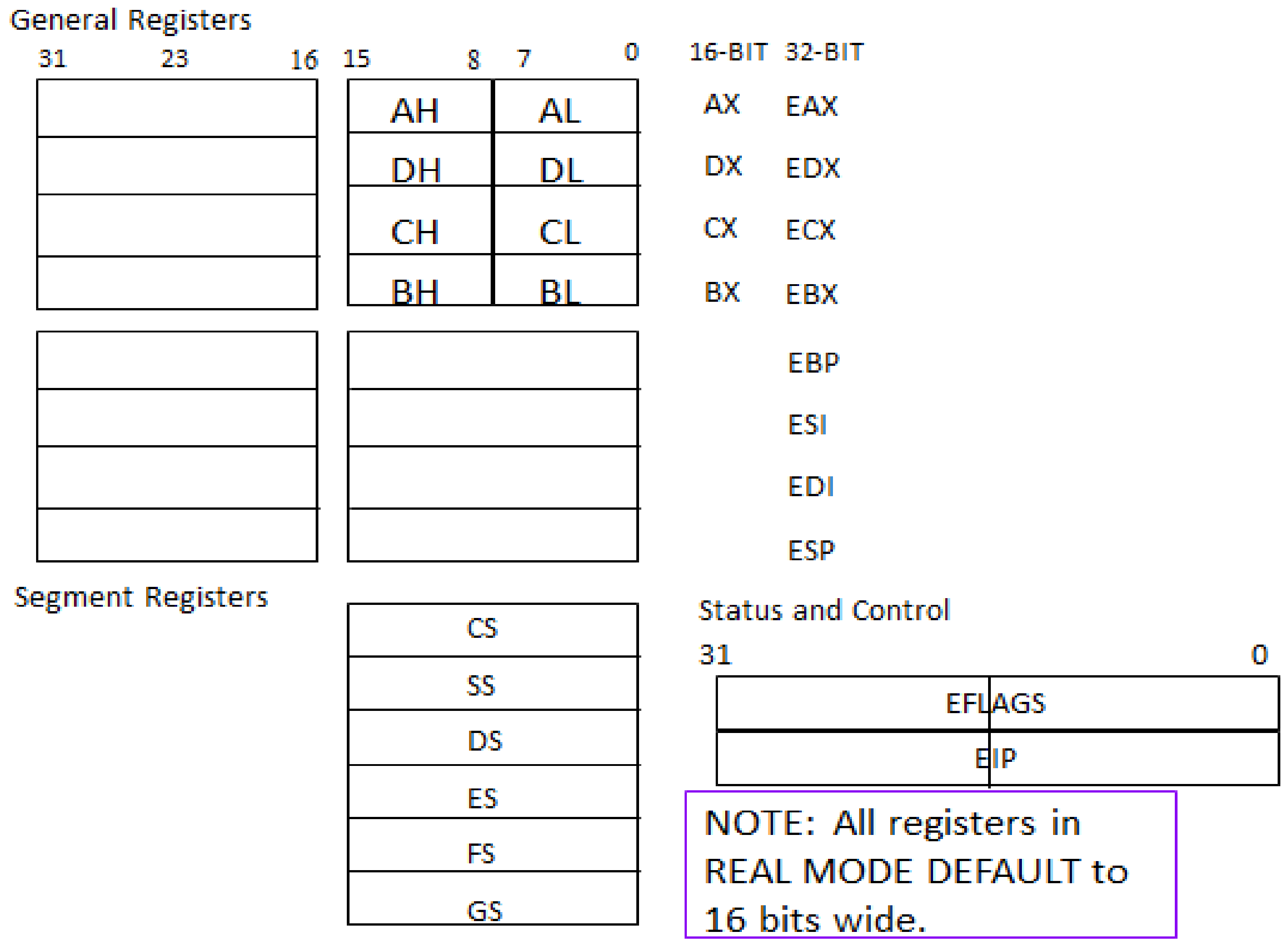
– The v-pipe can Execute simple integer instructions and the FXCH floating-point instructions.

– Pairing instructions in these two pipes enables the Pentium to operate on 2 instructions at the same time (Superscaler execution).

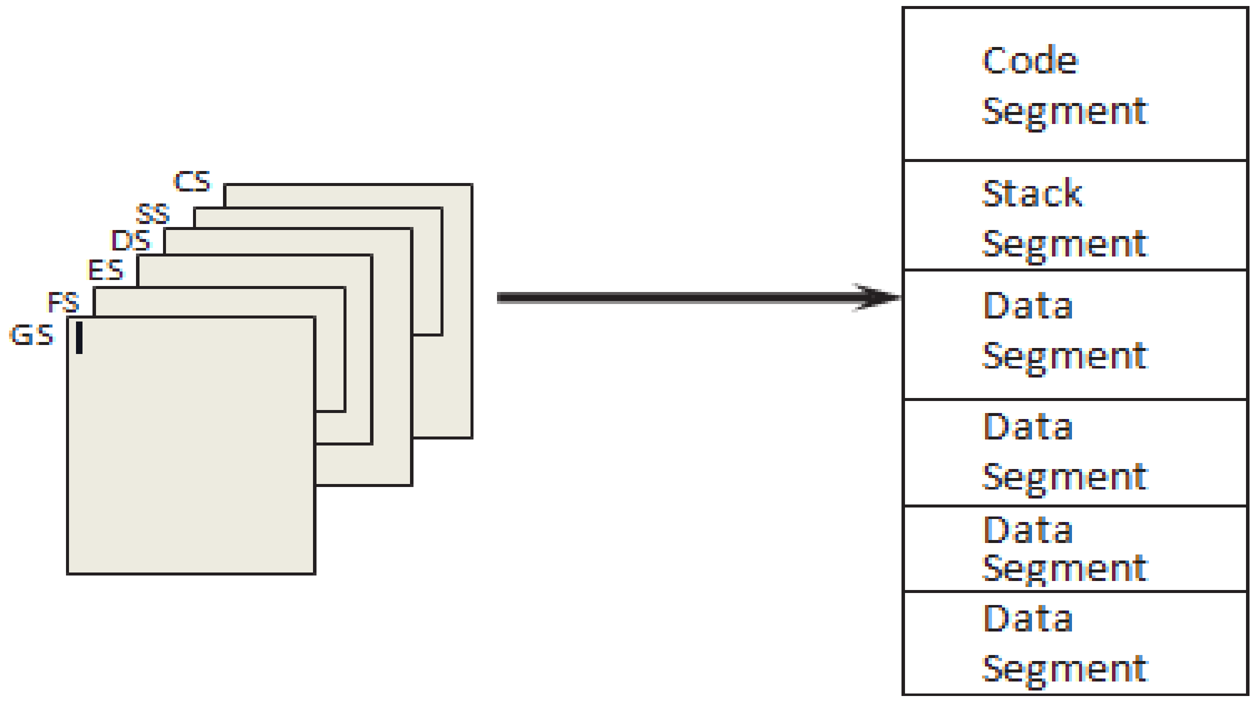
- The Control ROM unit has direct control over both pipelines.

– The Control ROM contains microcode which controls the sequence of operations that must be performed.

❖ Pentium Registers



❖ Segmented Addressing



❖ Pentium Registers (EFlags)

- The EFlags register is not a normal register but a collection of FLAG BITS which indicate the result of previous operations or the current state of the CPU.

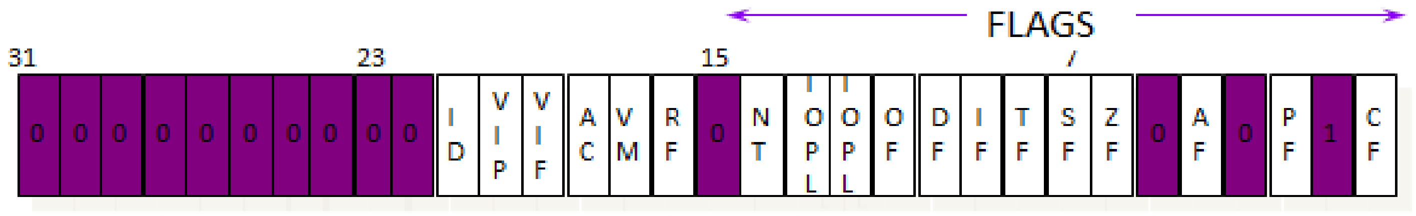
– A FLAG is just a flip-flop in the CPU that is SET (1) or RESET (0) [cleared] depending on the condition produced by an instruction .

– Some Flags indicate the condition produced by the previous instruction.

- e.g - Zero Flag: ZF=1 (True) if the result of the last arithmetic or logical operation was Zero.

– Some Flags are used to control certain operations.

- e.g. - IF (Interrupts Enabled); Trap Flag; Direction Flag.



ID	X	ID Flag (CPUID support)	DF	C	Direction Flag
VIP	X	Virtual Interrupt Pending	IF	X	Interrupt Enable Flag
VIF	X	Virtual Interrupt Flag	TF	X	Trap Flag
AC	X	Alignment Check	SF	S	Sign Flag
VM	X	Virtual 8086 Mode	ZF	S	Zero Flag
RF	X	Resume Flag	AF	S	Auxiliary Carry Flag
NT	X	Nested Task	PF	S	Parity Flag
IOPL	X	I/O Privilege Level	CF	S	Carry Flag
OF	S	Overflow Flag			

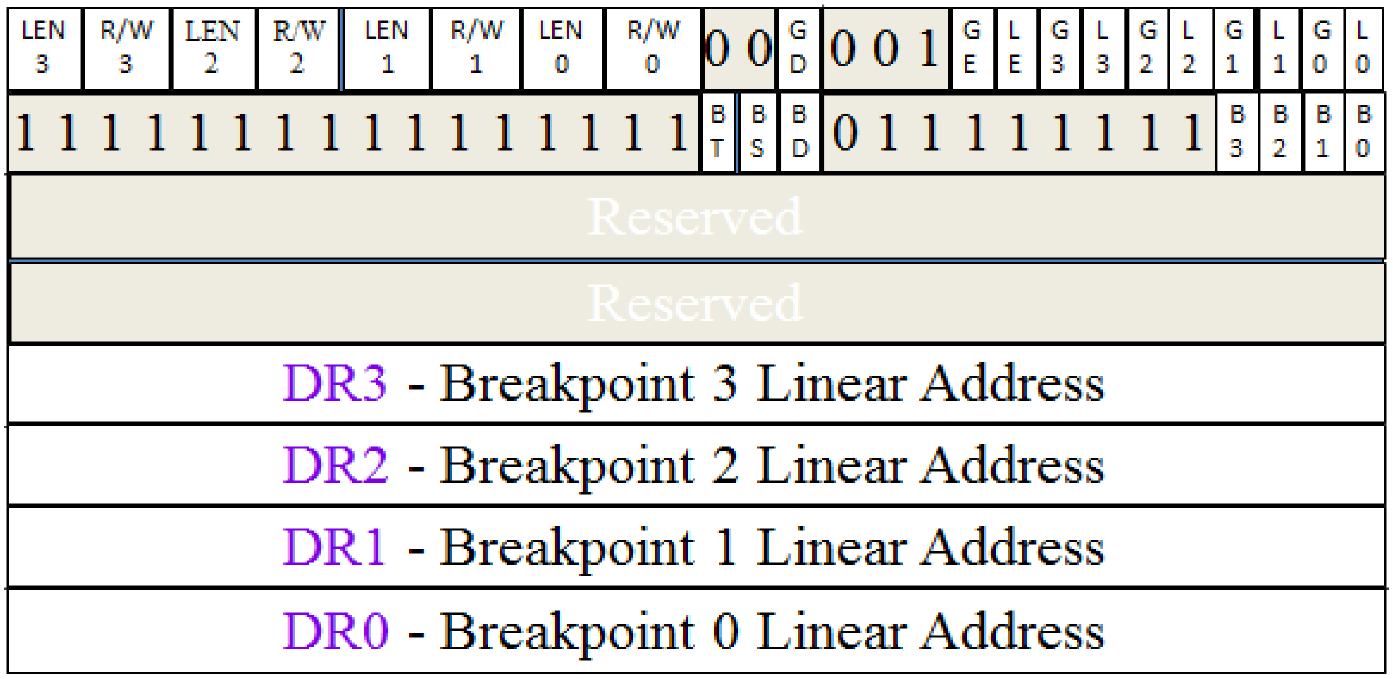
S = Status Flag

C = Control Flag

X = System Flag

🟪 Bit Positions shown as “0” or “1” are Intel reserved.

Pentium Registers (Debug)



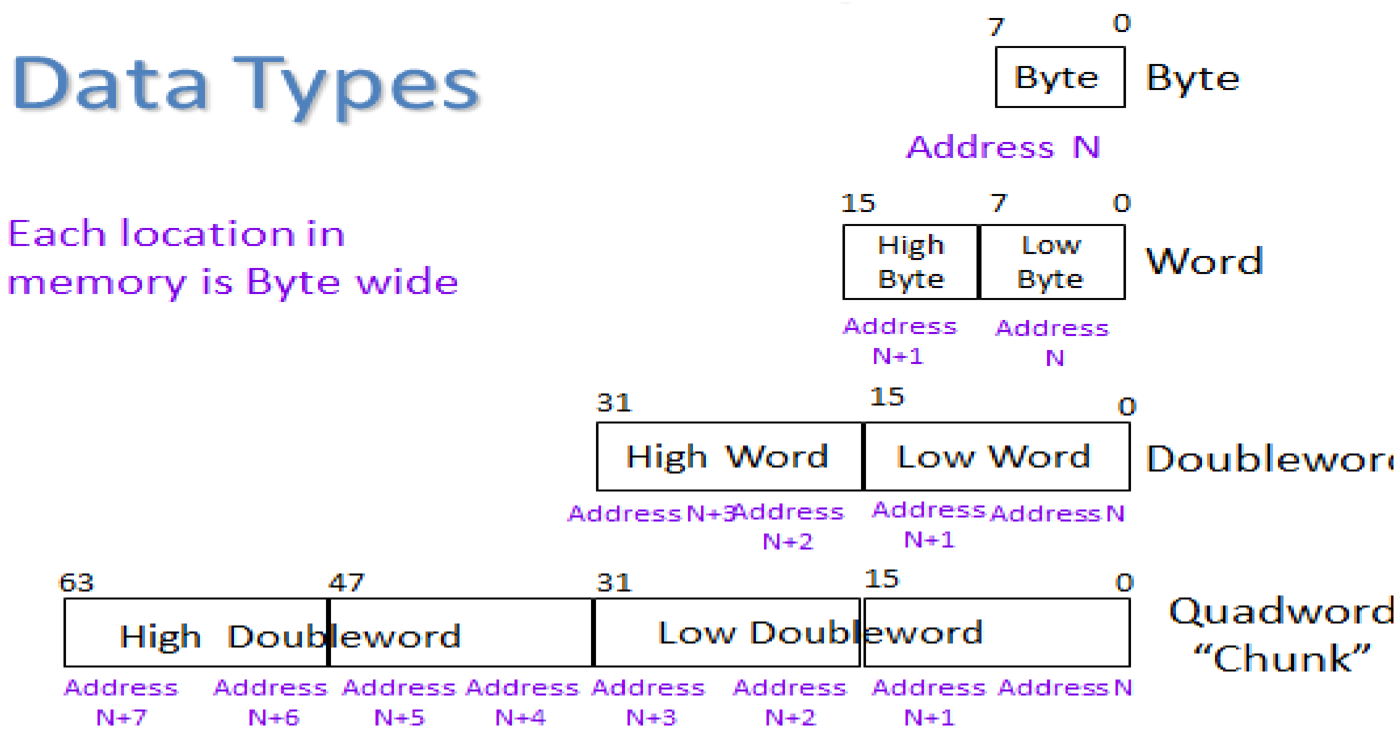
- The Debug Registers provide hardware support for setting breakpoints.
- You can define up to four breakpoints using Debug Registers (DR0 - DR3).
- The Debug Registers store the Linear Address.

  – The linear address is the address after the addition of the Segment Base & the Offset (w/o Paging).

  This is the Physical Address in REAL MODE.
- When the Pentium address matches an address in one of the Debug Registers, the Pentium issues a Debug Exception (INT 1).

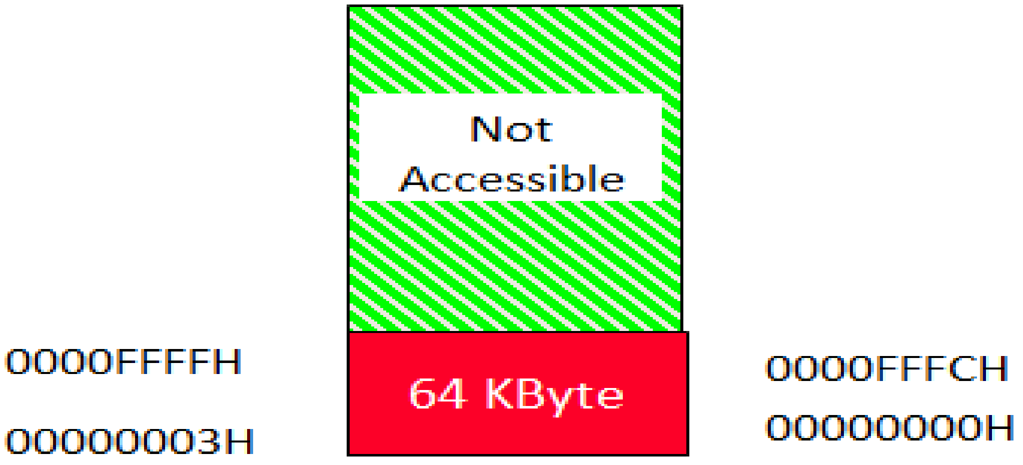
  –This feature is used with the ITP Debug Tool.

❖ Bus Description



❖ CPU Bus Description

- I/O Address Space is limited to 64 Kbytes (0000H-FFFFH).
- This limit is imposed by a 16 bit CPU Register.
  - A 16 bit register can store up to FFFFH (1111 1111 1111 1111 y).
- Which CPU Register limits I/O space to 64K



I/O Space

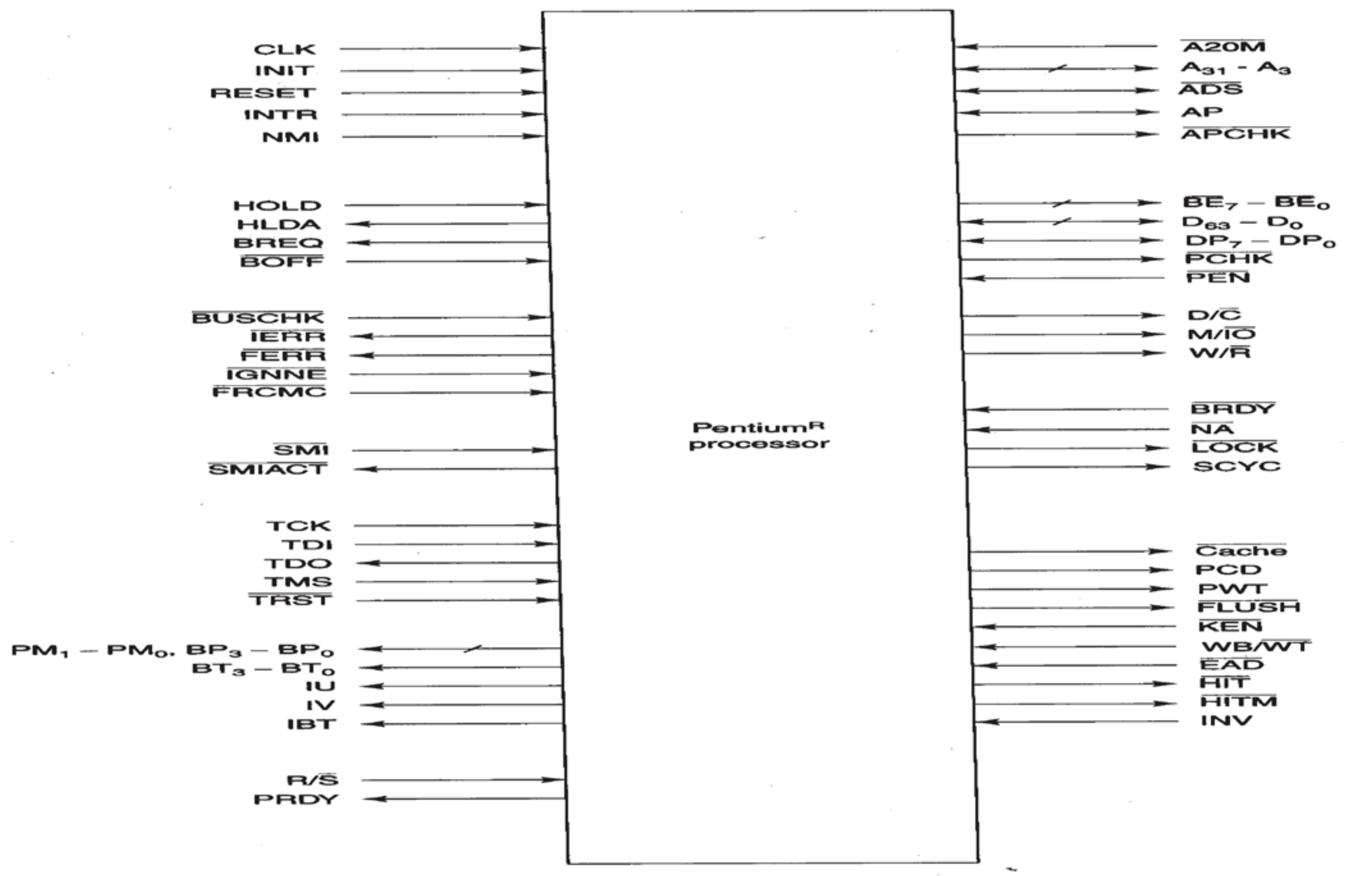


Figure 16.8 Block diagram of the Pentium® processor. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1995)

❖ Memory Subsystem

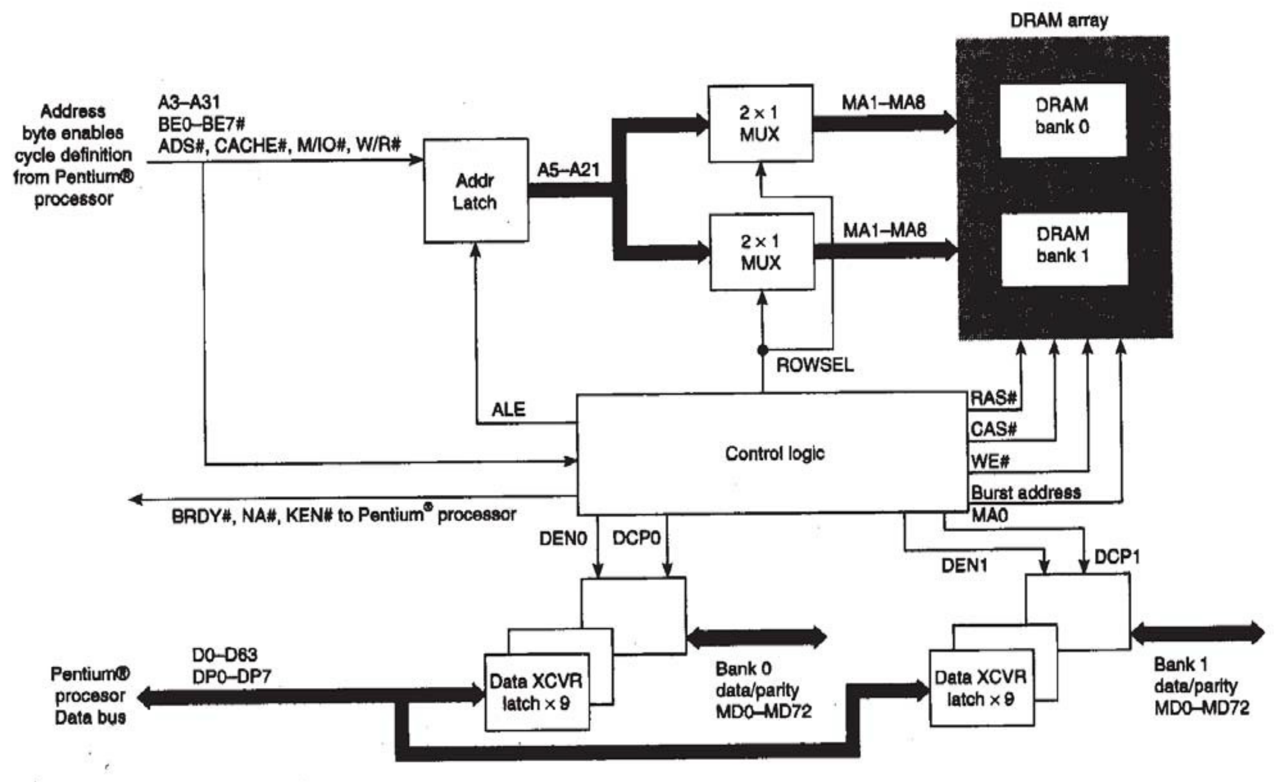

DRAM array
Address byte enables cycle definition from Pentium® processor
A3–A31
BE0–BE7#
ADS#, CACHE#, M/IO#, W/R#
Addr Latch
A5–A21
2 × 1 MUX
MA1–MA8
DRAM bank 0
DRAM bank 1
ROWSEL
ALE
Control logic
RAS#
CAS#
WE#
Burst address
MA0
BRDY#, NA#, KEN# to Pentium® processor
DEN0
DCP0
DEN1
DCP1
Pentium® procesor Data bus
D0–D63
DP0–DP7
Data XCVR latch × 9
Bank 0 data/parity MD0–MD72
Bank 1 data/parity MD0–MD72

Figure 16.9 Block diagram of a memory subsystem. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1995)

❖ Organization of the DRAM Array

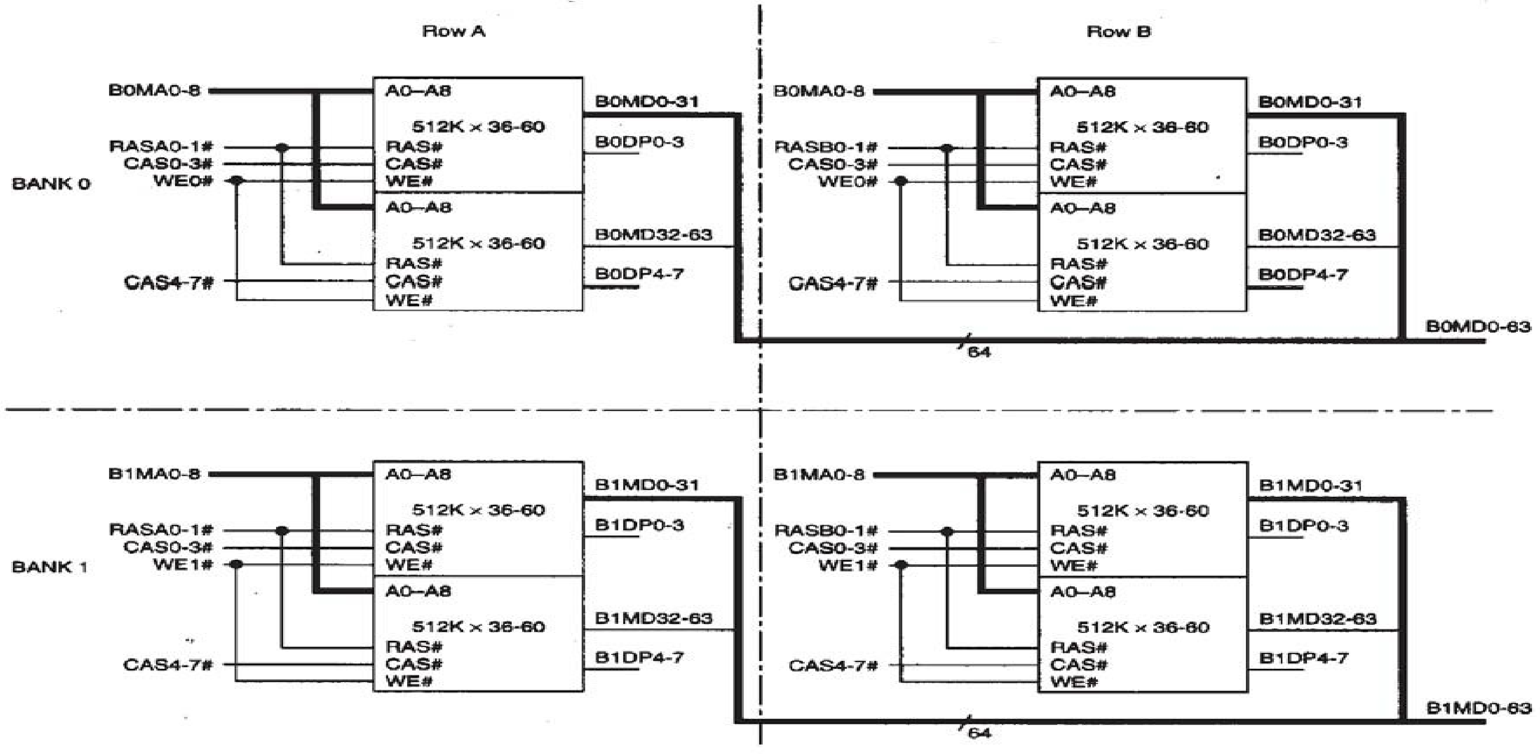


Figure 16.10 Organization of the DRAM array. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1995)

❖ RAS/CAS address MUX

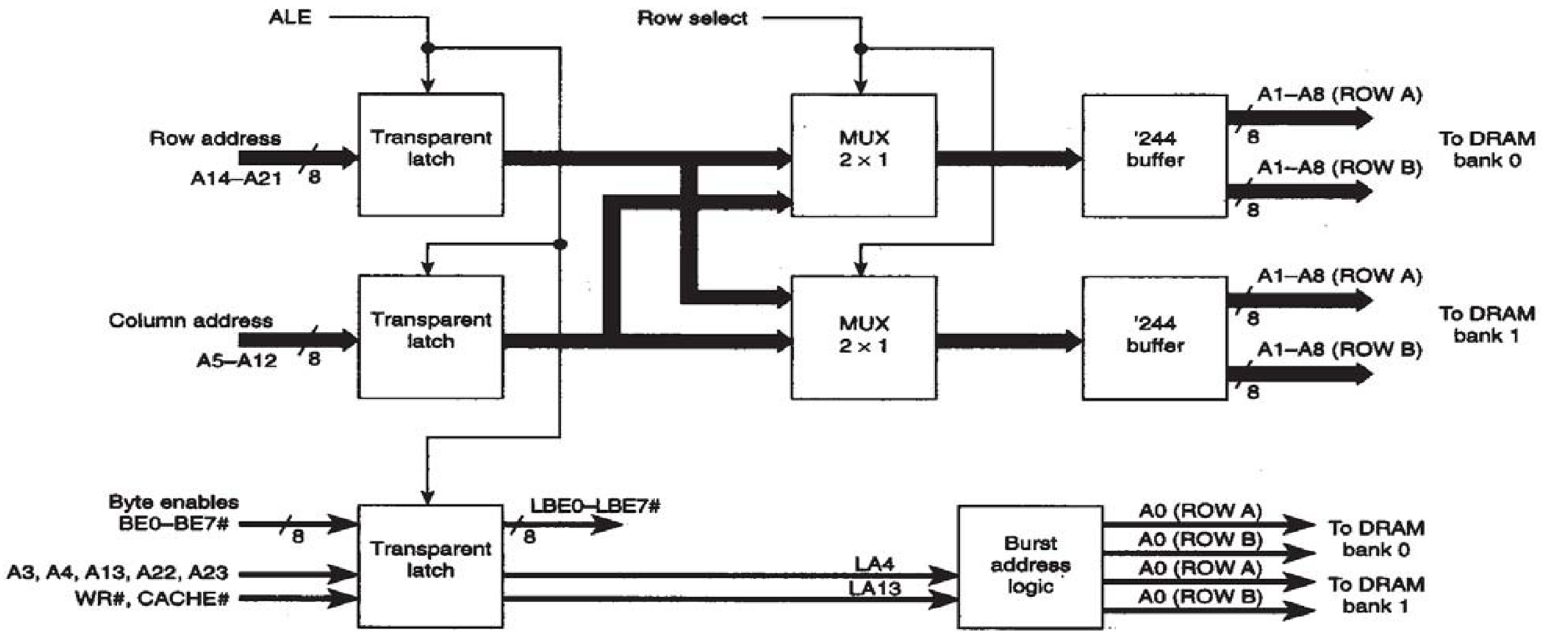


Figure 16.11 RAS/CAS address multiplexer circuits. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1995)

❖ Data Bus Transceiver Circuitry

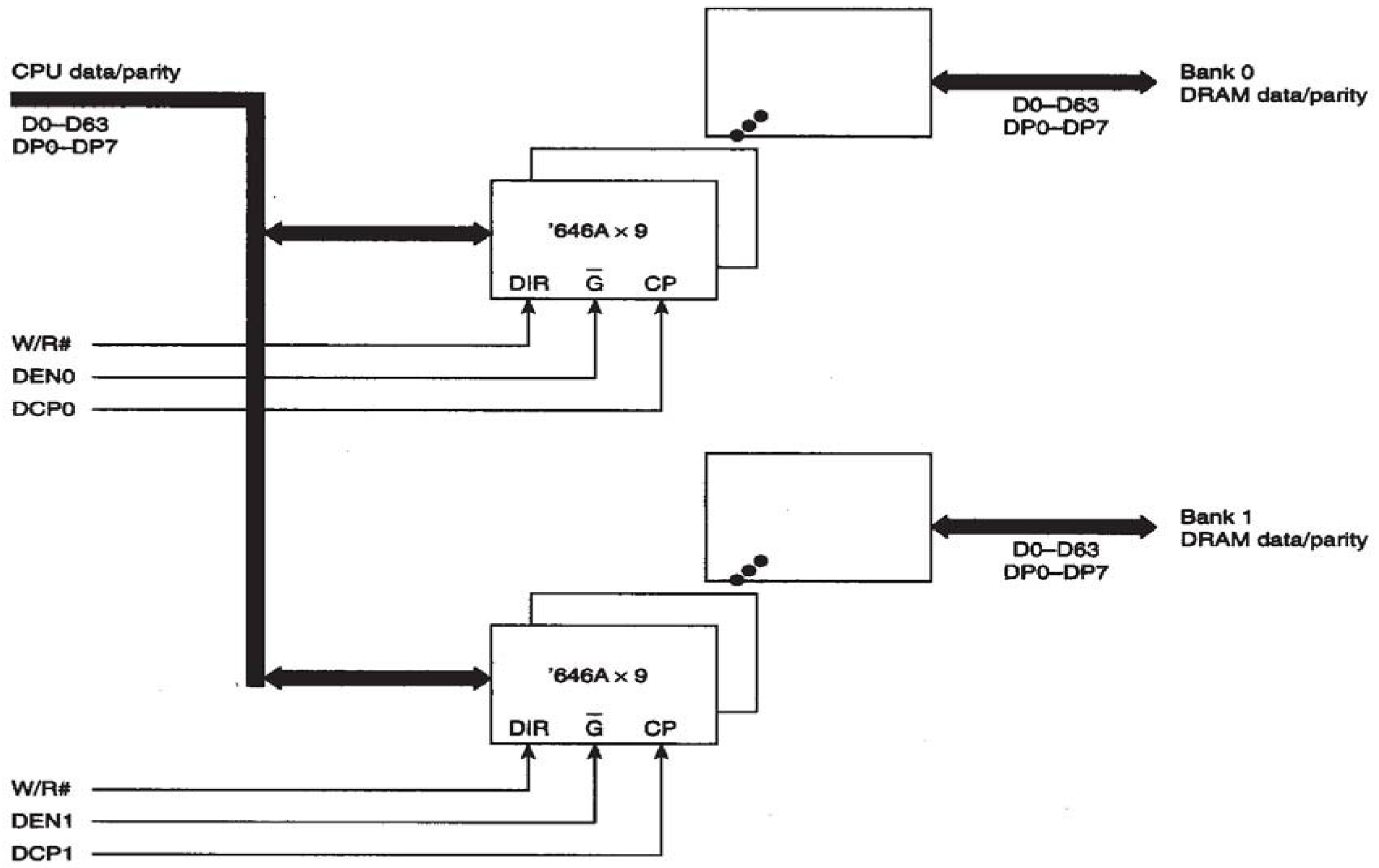


Figure 16.12 Data bus transceiver circuits. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1995)

❖ On-Chip Cache

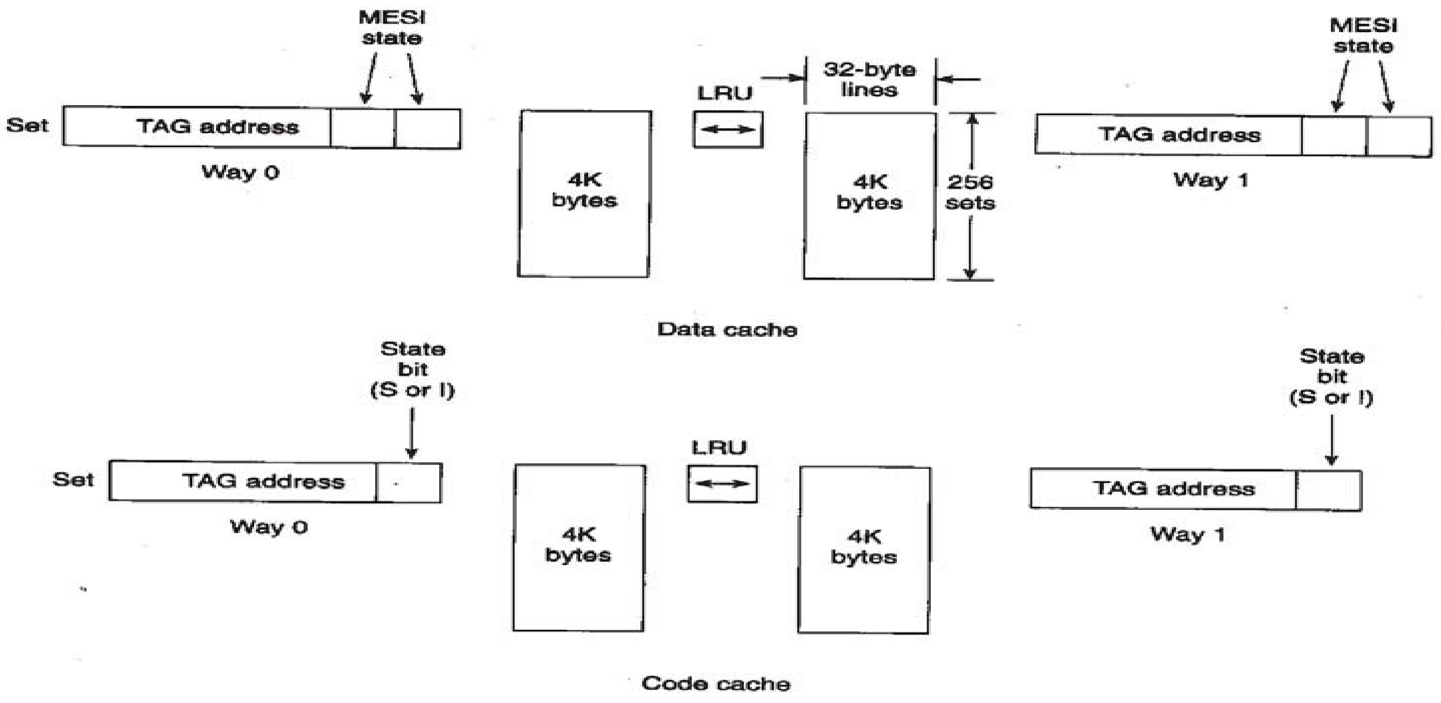


Figure 16.17 Organization of the on-chip cache of the Pentium^R processor. (Reprinted by permission of Intel Corp. Copyright/Intel Corp. 1995)

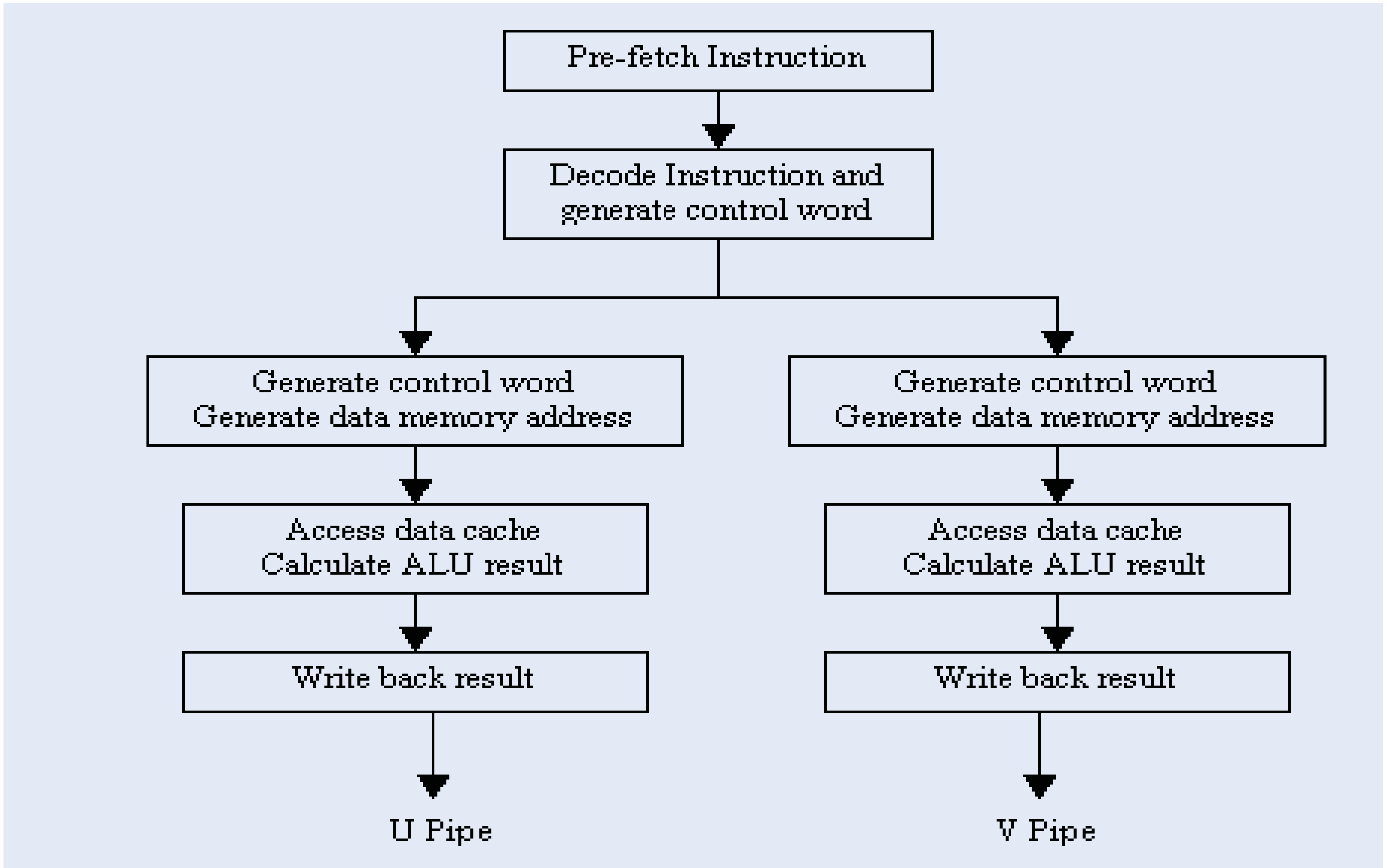

Pre-fetch Instruction
Decode Instruction and generate control word
Generate control word Generate data memory address
Generate control word Generate data memory address
Access data cache Calculate ALU result
Access data cache Calculate ALU result
Write back result
Write back result
U Pipe
V Pipe

Fig .2 Pentium pipeline stages

❖ The Intel® Pentium® Processor (1993)

- The introduction of the Intel Pentium processor added a second execution pipeline to achieve superscalar performance (two pipelines, known as u and v, together can execute two instructions per clock).
- The on-chip first-level cache doubled, with 8 KBytes devoted to code and another 8 KBytes devoted to data.
- The data cache uses the MESI protocol to
- support more efficient write-back cache in addition to the write-through cache previously used by the Intel486 processor.
- Branch prediction with an on-chip branch table was added to increase performance in looping constructs

❖ PROCESSOR FEATURES OVERVIEW

- The Pentium processor supports the features of previous Intel Architecture processors and
- provides significant enhancements including the following:
- Superscalar Architecture
- Dynamic Branch Prediction
- Pipelined Floating-Point Unit
- Improved Instruction Execution Time
- Separate Code and Data Caches.
- Writeback MESI Protocol in the Data Cache
- 64-Bit Data Bus
- Bus Cycle Pipelining
- Address Parity
- Internal Parity Checking
- Functional Redundancy Checking2 and Lock Step operation2
- Execution Tracing
- Performance Monitoring

❖ SUPER SCALAR EXECUTION

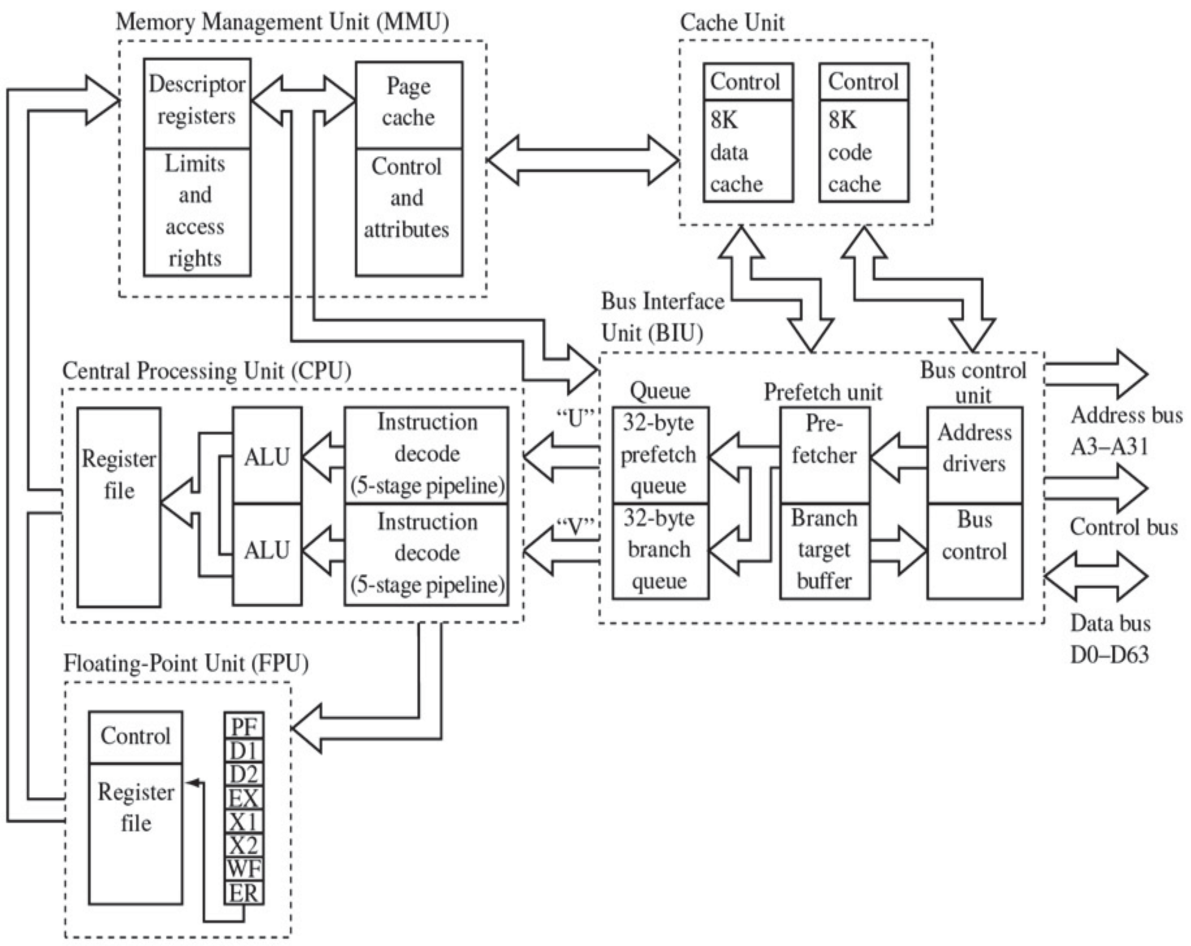
- A superscalar CPU architecture implements a form of parallelism called instruction level parallelism within a single processor. It therefore allows faster CPU throughput than would otherwise be possible at a given clock rate.
- A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor.
- Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier.

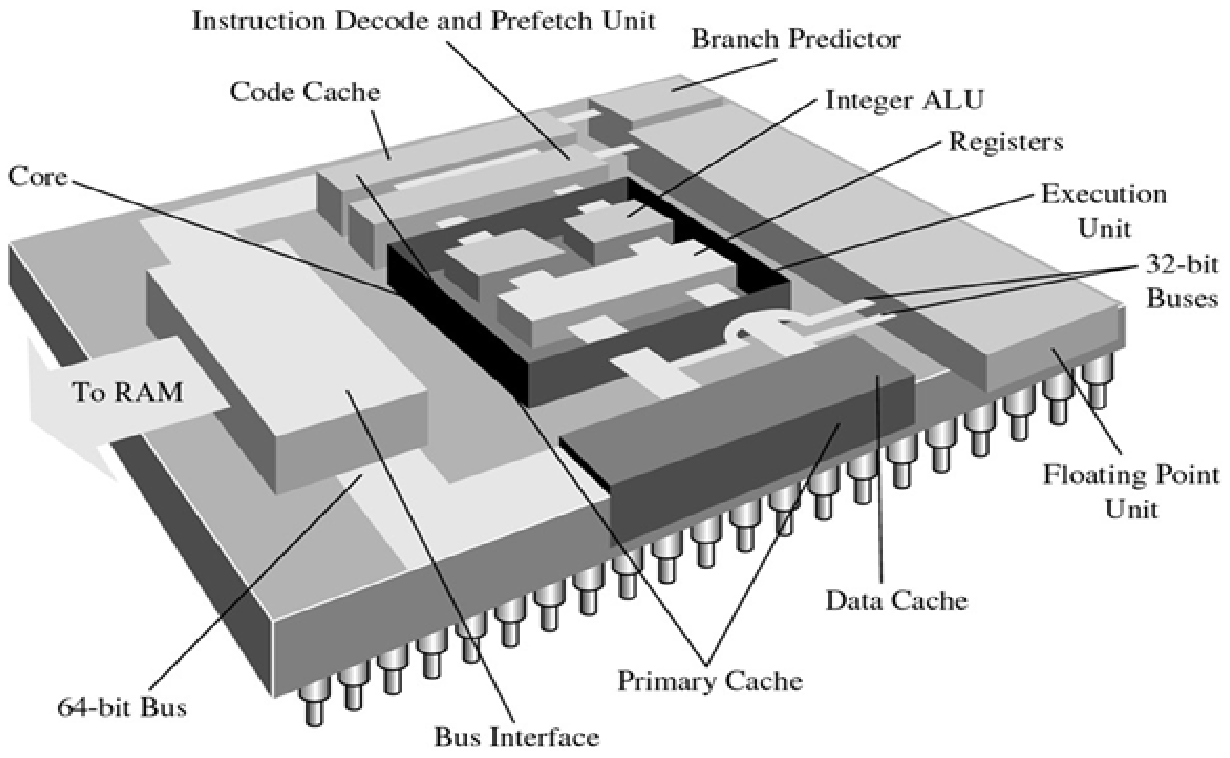
❖ Integer Instruction Pairing Rules

- The Pentium processor can issue one or two instructions every clock. In order to issue two instructions simultaneously they must satisfy the following conditions:

1. Both instructions in the pair must be “simple” as defined below Simple instructions are entirely hardwired; they do not require any microcode control and, in general, execute in one clock
2. There must be no read-after-write or write-after-write register dependencies between them.
3. Neither instruction may contain both a displacement and an immediate
4. Instructions with prefixes can only occur in the u-pipe.
5. Instruction prefixes are treated as separate 1-byte instructions. Sequencing hardware is used to allow them to function as simple instructions.

FIGURE 3-28 Processor model for the Pentium. The BIU supplies instructions to the CPU via two pipelines called the *u* and *v* pipes. In addition, two separate 8K data and code caches are provided.





❖ The U and V Pipes

- U and V pipes : dual five-stage pipelines
- Prefetcher and queue units provide paired instructions for U and V pipes
- U pipe : executes all Pentium instructions
- V pipe : executes only simple integer instructions (data is already in the CPU registers)
  -sorting of instructions is performed by the prefetcher
- Two pipelines and two ALUs -+ Pentium executes two instructions simultaneously (in one clock cycle).
  Y Condition : two instructions are simple and do not depend on each other – no data dependency.

❖ Superpipelined vs. Superscalar

- Superpipelining : divide the instruction execution pipeline into the smaller stages.
  [ex] 5-stage pipeline (80486, Pentium)
  -+ 12-stage (P6 processors)
- Superscalar : execute two or more instructions per clock cycle by using multiple execution units (include ALUs).
- [ex] Pentium executes two instructions simultaneously
  = 2-way superscalar
  Pentium II, III & Celeron : 3-way superscalar

❖ MMX (Multimedia Extension)

- MMX (Multimedia Extension) : provides 2 architectural enhancements over non-MMX Pentium
- CD57 instructions are added for multimedia (audio, video, and graphic data) applications.
- ® SIMD(Single-Instruction stream Multiple-Data stream) allows the same operation to be performed on multiple data items. Because many multimedia applications require large blocks of data to be manipulated, SIMD provides a significant performance enhancement.
- For general applications, 10~20% performance improved. For multimedia applications, nearly 70% improved.

Chapter no 3 RISC Architecture

❖Overview

- History of CISC and RISC
- CISC and RISC
  -DPhilosophy
  -DAttributes and disadvantages
- Summation

❖History of RISC/CISC

- 1950s IBM instituted a research program
- 1964 Release of System/360
- Mid-1970s improved measurement tools demonstrated on CISC
- 1975 801 project initiated at IBM's Watson Research Center
- 1979 32-bit RISC microprocessor (801) developed led by Joel Birnbaum
- 1984 MIPS developed at Stanford, as well as projects done at Berkeley
- 1988 RISC processors had taken over high-end of the workstation market
- Early 1990s IBM's POWER (Performance Optimization With Enhanced RISC) architecture introduced w/ the RISC System/6k
- AIM (Apple, IBM, Motorola) alliance formed, resulting in PowerPC

❖Hybrid architecture(CISC and RISC Convergence)

- From last 30 year debate on two microprocessor architeuture (risc & cisc) o decide which one is better for processor
- Now a days processor has been launched which uses combination of both (RISC & CISC)architecture.
- In hybrid architecture the processor designs use only the best features of RISC & CISC and mix them in hybridemicro-
- RISC architecture is simple & efficient One group of designer support RISC architecture design
- CISC have low burden on compiler developers and wide availability of existing software, another group of designer support CISC architecture design
- The hybrid (RISC & CISC architecture) uses decoder which convert cisc instruction into risc prior to execution (fast exe)
- RISC architecture is better performance enhancment feature such as pipelining & branch prediction
- Now a days modern RISC processor have become more CISC like supporting more function & more instruction than old CISC design
- CISC arcitecture have more instruction
- ,more application may run faster such as multimedia app.,telecommuication encoding/decoding ,image conversion & video processing

- CISC and RISC Convergence

- State of the art processor technology has changed significantly since RISC chips were first introduced in the early '80s. Because a number of advancements are used by both RISC and CISC processors, the lines between the two architectures have begun to blur. In fact, the two architectures almost seem to have adopted the strategies of the other. Because processor speeds have increased, CISC chips are now able to execute more than one instruction within a single clock. This also allows CISC chips to make use of pipelining. With other technological improvements, it is now possible to fit many more transistors on a single chip.

- This gives RISC processors enough space to incorporate more complicated, CISC-like commands. RISC chips also make use of more complicated hardware, making use of extra function units for superscalar execution. All of these factors have led some groups to argue that we are now in a "post-RISC" era, in which the two styles have become so similar that distinguishing between them is no longer relevant. However, it should be noted that RISC chips still retain some important traits. RISC chips stricly utilize uniform, single-cycle instructions. They also retain the register-to-register, load/store architecture. And despite their extended instruction sets, RISC chips still have a large number of general purpose registers.

❖ Basic features of RISC

- Single cycle L/S instruction execution
- (faster,low overhead)
- The addressing mode in RISC processor are lower & all the operation are takes place within the CPU
- Hardwired control ; vastly reduced chip complexity
- more registers/less MEM references(only LOAD & STORE instn) more efficient instruction pipeline.
- Fixed INST format,fewer instructions can be easily decoded
- DMEM accesses are not tightly bound to INSTS
- Few addressing mode are used e.g. register,register direct,reg. indirect,displacement

❖ Avantages of RISC Processor

- A new microprocessorcan be developed & tested more quickly which can take advantage of other technological developments leading to the discussion between performance & generation.
- It is easier to develop a code with less instruction set for application & operating system program
- Here more freedom to decide how to use the scope of a microprocessor
- Higher level language compilers are used to generate more efficient code in Risc(small set of instn)
- Super scaler RISC processor 2 to 4 time faster than CISC because of pipelined feature.
- The RISC processor requires less chip space of additional function such as mem management unit & floating point arithmetic unit
- Smaller chip size lower the per chip cost

- All inst executed in single clock cycle hence have faster execution time.
- Output regs.- the programmer can store arguments in these regs while calling a function
- Only one window is active (visible) at any time
- When window run out, then it dumps window contents onto the stack which is expensive and also 64-byte stack space must be reserved for each call at all times
- Also it is impossible to predict when the register window will overflow/underflow hence performance will be unpredictable.

❖ Design issues of RISC processor

❖ 1.Register window

- The reg window in RISC processor is use to improve performance of common operation & procedure call
- In SPARCprocessor,an instruction can access 32 regs. i.e 8 global reg & 24 register window at any time.
- Register window contain 16 register (8 in & 8 local reg.)
- 8 in reg. are the consective reg. & addressable from the current window as out register
- When a procedure or sub routine is called the reg. window is shifted by 16 reg which hides the old input reg & old local reg which make the old output reg as new input regs.
- Input regs. Are used to pass arguments to procedure or subroutine.
- Local regs. used to store local data

❖ 2 Pipelining in RISC

- Multiple instruction are overlapped during execution which take advantages of parallelism
- Now days pipelining is major tech to increase the speed
- Five stage pipelining in RISC
- 1.intsruction fetch(IF)
- Instruction decode-ID-(read the reg source from reg file)
- Ex(execution /effective addr. Cycle) –
  addres calculation & execution combine in one cycle
- MEM address– read from mem usin E.A computed in previous cycle or data from second reg read from reg file is written into mem using E.A
- WB(write back)—write result into reg file

❖ 3 Single cycle instruction execution in RISC

(instruction latency in RISC designe)

- We can pipeline the execution with almost no
- changes by simply starting a new instruction on each clock cycle
- Each of the clock cycle from previous section becomes a pipe stage : acycle in the pipeline.
- Each instruction take five clock cycle to complete ,during each clock cycle the hardware will initiate a new instruction & will be executed some part of five different instructions.
- The pipelining of instruction execution increase the CPU instruction throughput but it does not reduce the execution time of individual instruction

- Practically the execution time of each instruction is increase due to overhead in the control of the pipeline.
- No single instruction run faster,but still program run fasterwith lower total execution time.
- Pipeline overhead arises from the combination of pipeline reg delay & clock skew.
- There is a situation which prevents the next instruction in the instruction stream from executing its designated clock cycle .(which degrade the pipeline performance from ideal performance.

❖ 4 Performance issues in pipelining

- Pipeling of inst execution increase the cpu speed (throughput) but it it does not reduce the time of individual instruction.
- The execution time of each instruction is increased due overhead in the control of the pipeline (practically)
- The execution time of each instruction does not decrease by pipelining the instruction but limit the depth of a pipeline
- The limit generates from imbalance among the pipe stages & pipeline overhead with the limitation generating from pipeline latency.
- Imbalance in the pipe stages decreases performance of cpu as the clock can not run faster for the slowest pipeline stage.
- ,Setup time addede pipeline regs , is time that a reg input must be stable before the clock signal that triggers a write occures, plus propagation delay to the clock cycle.
- Clock skew is the max delay between the clock arrives at any two regs & put the lower limit on the clock cycle.
- The pipelining is not useful when the clock cycle is small a compared to the sum of the clock skew and latch overhead,hence no time left in the cycle for useful during the execution of an instruction.

❖5 Data dependency issues

- RISC processor perform operation more than one clock cycle per instruction and can be occasionally stall due to the result of data dependencies and branch instructions.
- when the execution of instruction depends on the result of a previous instruction, then the data dependency occures
- A particular instruction may need data in a register which is not available in register,hence next instruction has to wait till the data it needs is stored in the reg.
- This situation is called as stalling(available) of pipelining where no of empty instruction are inserted into the pipeline

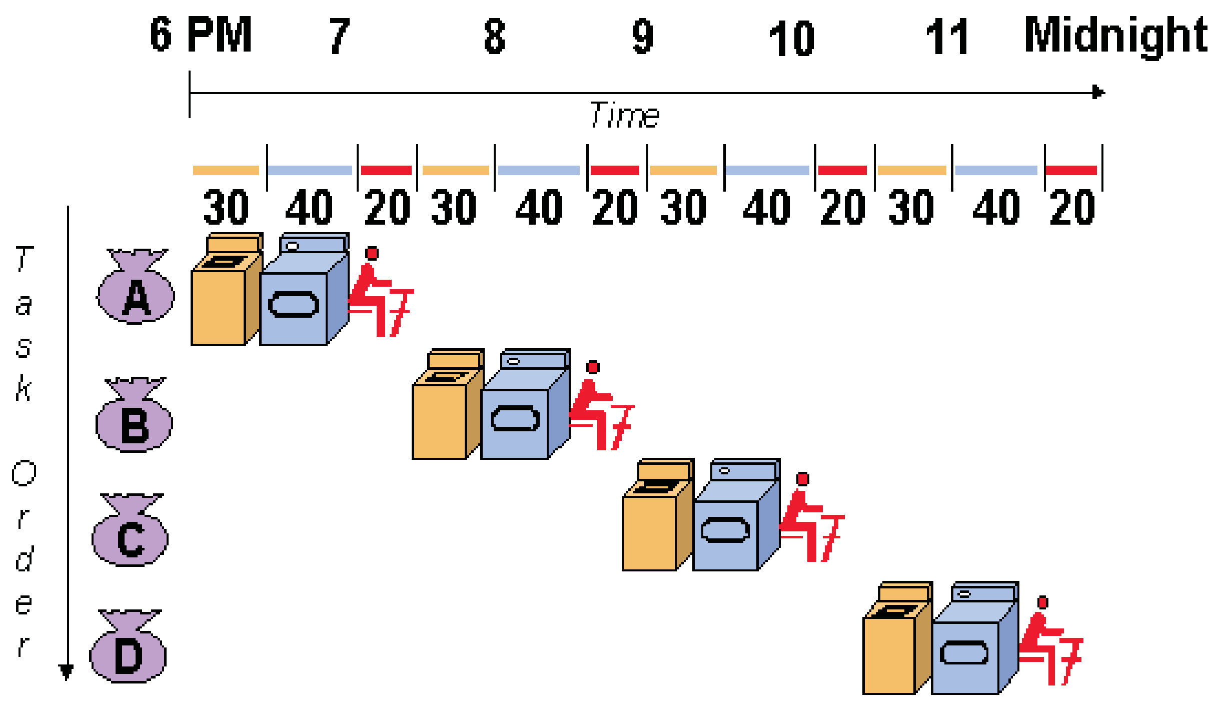
❖6 Features of Sun Ultra SPARC

- Scalable processor architectured developed by SUN Microsystem. & design by Texas Instruments with SPARC V9 instruction set architecture.
- It is first microprocessor designed by SUN Microsystem in 1995
- Ultra SPARC is a four super scaler microprocessor that execute instruction in in-order nine stage pipeline
- It has 2 level of cache(primary & secondary)
- Two primary cache each of 16kb(one for instruction & one for data)
- Instruction in instruction cache r pre-decoded
- It has external secondary cache (512KB to 4 MB)
- All arithematic instruction are reg to reg & 64-bit
- Precision i.e 64 bit integer mul & div instruction 64-bit virtual memory address
- Super scaler execution (nine stage pipeline & 4inst per cycle) to minimize latency.
- Support reg windows

❖ Pipelining

❖ RISC Pipelines

- A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:
  - fetch instructions from memory
  - read registers and decode the instruction
  - execute the instruction or calculate an address
  - access an operand in data memory
  - write the result into a register



❖ RISC Disadvantages

- There is still considerable controversy among experts about the ultimate value of RISC architectures. Its proponents argue that RISC machines are both cheaper and faster, and are therefore the machines of the future.
- However, by making the hardware simpler, RISC architectures put a greater burden on the software. Is this worth the trouble because conventional microprocessors are becoming increasingly fast and cheap anyway?

❖ CISC versus RISC

CISC	RISC
• Emphasis on hardware	• Emphasis on software
• Includes multi-clock complex instructions	• Single-clock, reduced instruction only
• Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	• Register to register: "LOAD" and "STORE" are independent instructions
• Small code sizes, high cycles per second	• Low cycles per second, large code sizes
• Transistors used for storing complex instructions	• Spends more transistors on memory registers

❖ Summation

- As memory speed increased, and high-level languages displaced assembly language, the major reasons for CISC began to disappear, and computer designers began to look at ways computer performance could be optimized beyond just making faster hardware.

- One of their key realizations was that a sequence of simple instructions produces the same results as a sequence of complex instructions, but can be implemented with a simpler (and faster) hardware design. (Assuming that memory can keep up.) RISC (Reduced Instruction Set Computers) processors were the result.

- CISC and RISC implementations are becoming more and more alike. Many of today’s RISC chips support as many instructions as yesterday's CISC chips. And today's CISC chips use many techniques formerly associated with RISC chips.

- To some extent, the argument is becoming moot because CISC and RISC implementations are becoming more and more alike. Many of today's RISC chips support as many instructions as yesterday's CISC chips. And today's CISC chips use many techniques formerly associated with RISC chips.