Unit 2

Microprocessor Operational and addressing modes

Features of 80286

- It is 16 bit MP but faster than 8086
- The 80286 CPU, with its 24-bit address bus is able to address 2^{24}=16 Mbytes of physical memory.
- The clock frequencies are 4MHz , 6MHz, 8MHz , 10 MHz and 12.5 MHz .
- 80286 is upwardly compatible with 8086 in terms of instruction set.

- Enhanced instruction set
- It is 4 stage pipeline microprocessor .
- Data protection or unauthorized access prevention is the important aspect of Memory management unit (MMU)
  - Segmented memory

- **80286 works in 2 operating modes:**
- **Real address mode**
  - In RM 80286 just acts as fast 8086.(addresses 1Mb memory)
  - 8086 program can be executed without modification in 80286
  - MMU and protection mechanisms are disabled in this mode.
- **Protected virtual address mode(PVAM)**
  - 80286 can address 16Mb of memory
  - Supports multitasking –run several program at the same time.
  - Works with its MM and protection capabilities with advanced instruction set.
  - it can address upto 1GB virtual memory .

Real addressing mode:

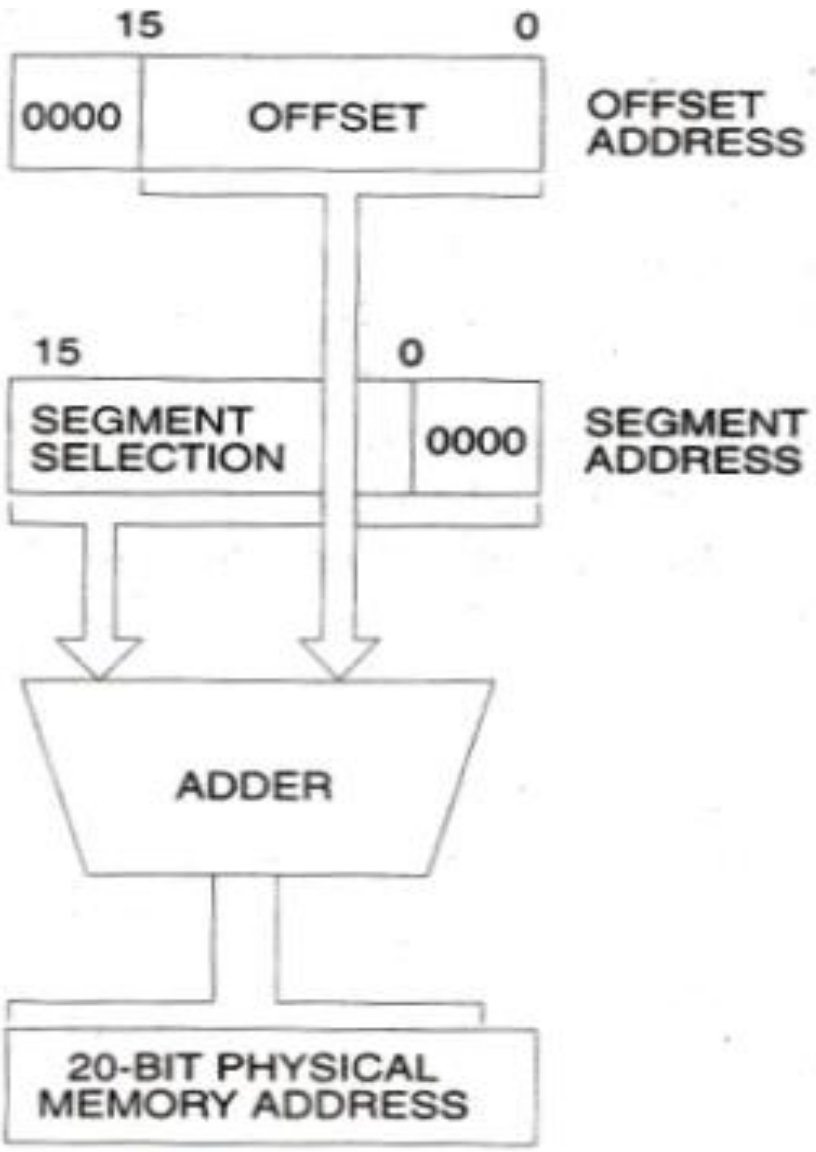


Figure: Real Address Mode Address Calculation

- 20 bit physical address is formed in the same way as that in 8086
- The contents of segment registers are used as segment base addresses.
- The other registers, depending upon the addressing mode, contain the offset addresses.
- PA = CS *10h + offset(IP)

Protected Virtual Address Mode (PVAM)

- 80286 is the first processor to support the concepts of virtual memory and memory management.

- The concept of VM is implemented using Physical memory that the CPU can directly access and secondary memory that is used as a storage for data and program, which are stored in secondary memory initially.

- The Segment of the program or data required for actual execution at that instant is fetched from the secondary memory into physical memory.

- After the execution of this fetched segment, the next segment required for further execution is again fetched from the secondary memory, while the results of the executed segment are stored back into the secondary memory for further references.

- This continues till the complete program is executed

- During the execution the partial results of the previously executed portions are again fetched into the physical memory, if required for further execution.

- The procedure of fetching chosen program segments or data from the secondary storage into the physical memory is called **swapping.**

- Procedure of storing back the partial results back on to secondary storage is called **unswapping.**
- The virtual memory is allotted per task.

- The 80286 is able to address 1G byte (2^{30} bytes) of virtual memory per task.

- The complete virtual memory is mapped on to the 16Mbyte physical memory.

- If a program larger than 16Mbyte, is stored on the hard disk and is to be executed, if it is fetched in terms of data or program segments of less than 16Mbyte in size into the program memory by swapping sequentially as per sequence of execution.

- Whenever the portion of a program is required for execution by the CPU, it is fetched from the secondary memory and placed in the physical memory is called *swapping in* of the program.

- A portion of the program or important partial results required for further execution, may be saved back on secondary storage to make the Physical Memory free for further execution of another required portion of the program is called *swapping out* of the executable program.

- 80286 uses the 16-bit content of a segment register as a selector to address a descriptor stored in the physical memory.

- The descriptor is a block of contiguous memory locations containing information of a segment, like segment base address, segment limit, segment type, privilege level, segment availability in physical memory, descriptor type and segment used for another task.

Physical address calculation in PVAM:

Physical address calculation in PVAM:

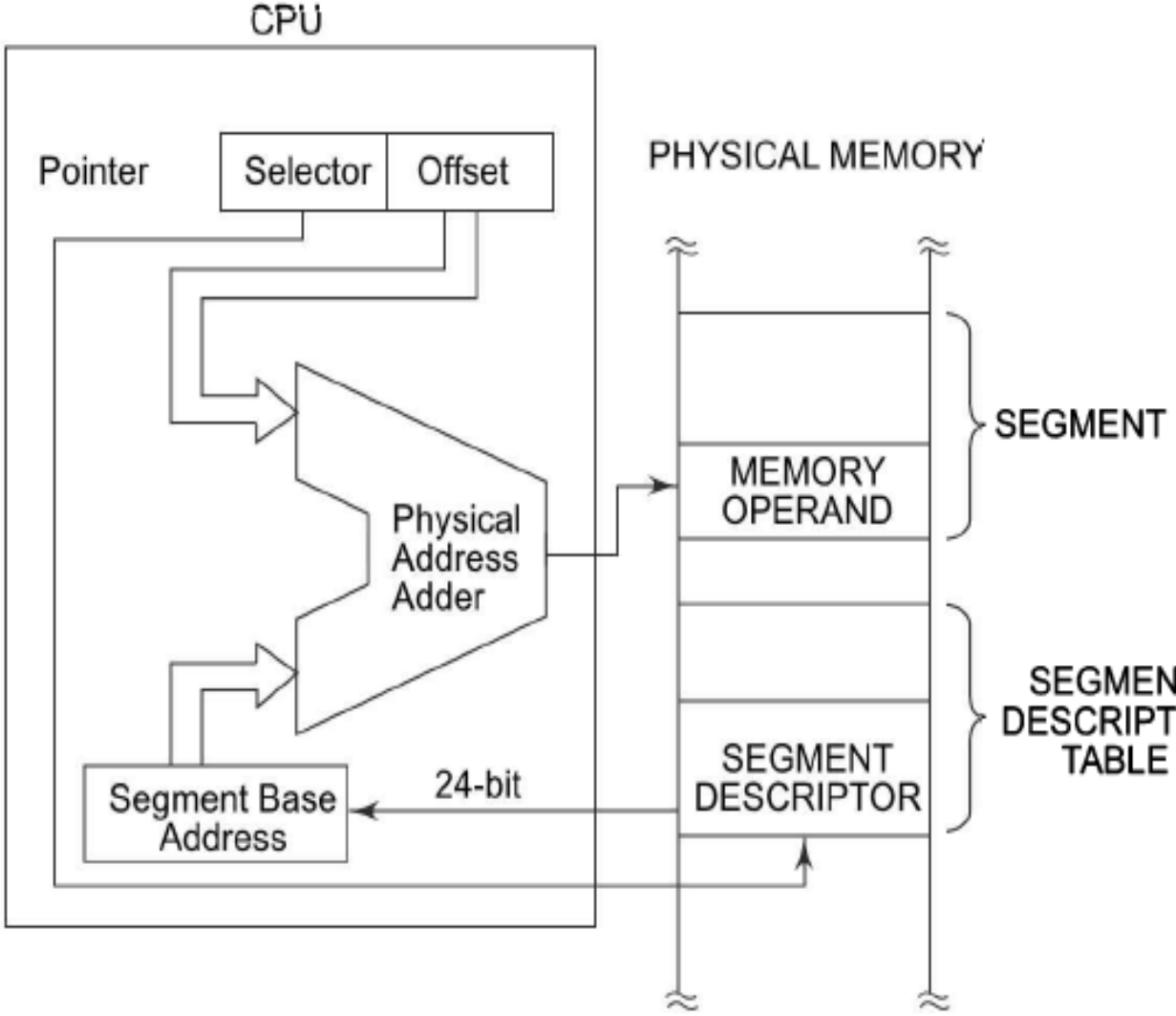


Fig. *Physical Address Calculation in PVAM (Intel Corp.)*

- Segment base address is a 24 bit , that addresses the first location in that segment.
- This 24 bit segment base address is added with 16 bit offset to calculate 24 bit physical address.
- Max segment size will be 64Kb ,as offset is of 16 bits.
- Descriptors are automatically referred to by the CPU when a segment register is loaded with a selector.

Program-Invisible Registers

- Cache was introduced in 80286 to minimize the time required for fetching the frequently required descriptor information from the main memory.

- Caching maintains the most frequently required data for execution in a high speed memory called cache memory.

- A 6-byte segment descriptor cache is assigned to each of the four segments i.e. CS,DS,SS and ES.

- A segment descriptor is automatically loaded in segment descriptor cache , whenever the associated segment is loaded with a selector.

- Once a cache register is loaded , all the information regarding the segment is obtained from the cache register, instead of referring to the main memory for the descriptor again and again.

- These cache registers are not available for programming

- They automatically change when segment register is loaded

- The program-invisible registers are not directly addressed by software so they are given this name .

  These registers control the microprocessor when operated in protected mode.

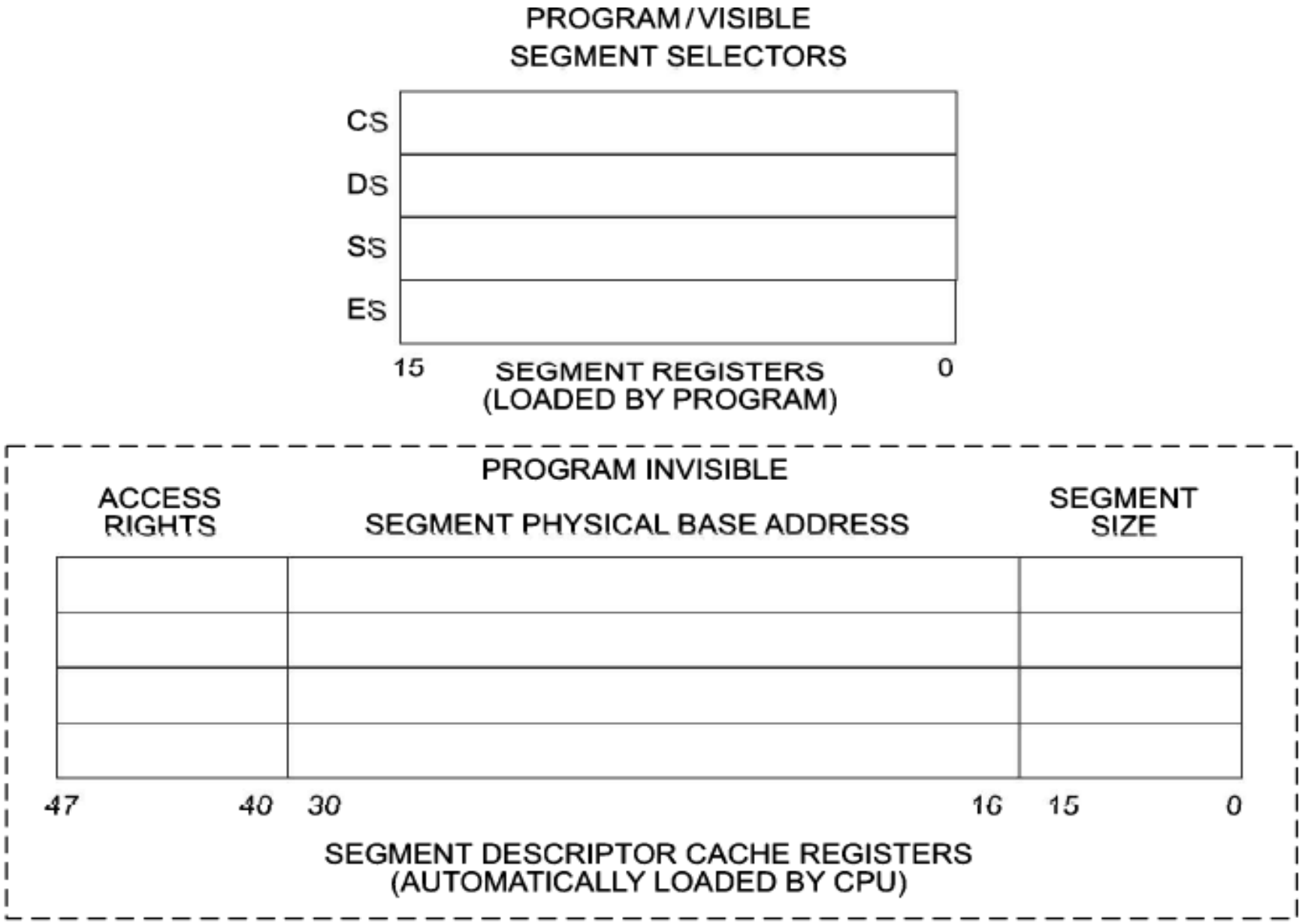


Fig. ***Descriptor Cache Register and their Formats (Intel Corp.)***

- Each of the segment registers contains a program-invisible portion used in the protected mode.

- The program-invisible portion of these registers is often called cache memory because cache is any memory that stores information.

- The program-invisible portion of the segment register is loaded with the base address, limit, and access rights each time the number segment register is changed.

- When a new segment number is placed in a segment register, the microprocessor accesses a descriptor table and loads the descriptor into the program-invisible portion of the segment register.

FLAT MODE MEMORY

- Pentium-based computer (Pentium 4 or Core2) that uses the 64-bit extensions uses a flat mode memory system.

- A *flat mode memory system is one in which there is* ***no segmentation****.*

- The address of the first byte in the memory is at 00 0000 0000H and the last location is at FF FFFF FFFFH (address is 40-bits).

FLAT MODE MEMORY

- The flat model does not use a segment register to address a location in the memory.

- The CS segment register is used to select a descriptor from the descriptor table that defines the access rights of **only a code segment**.

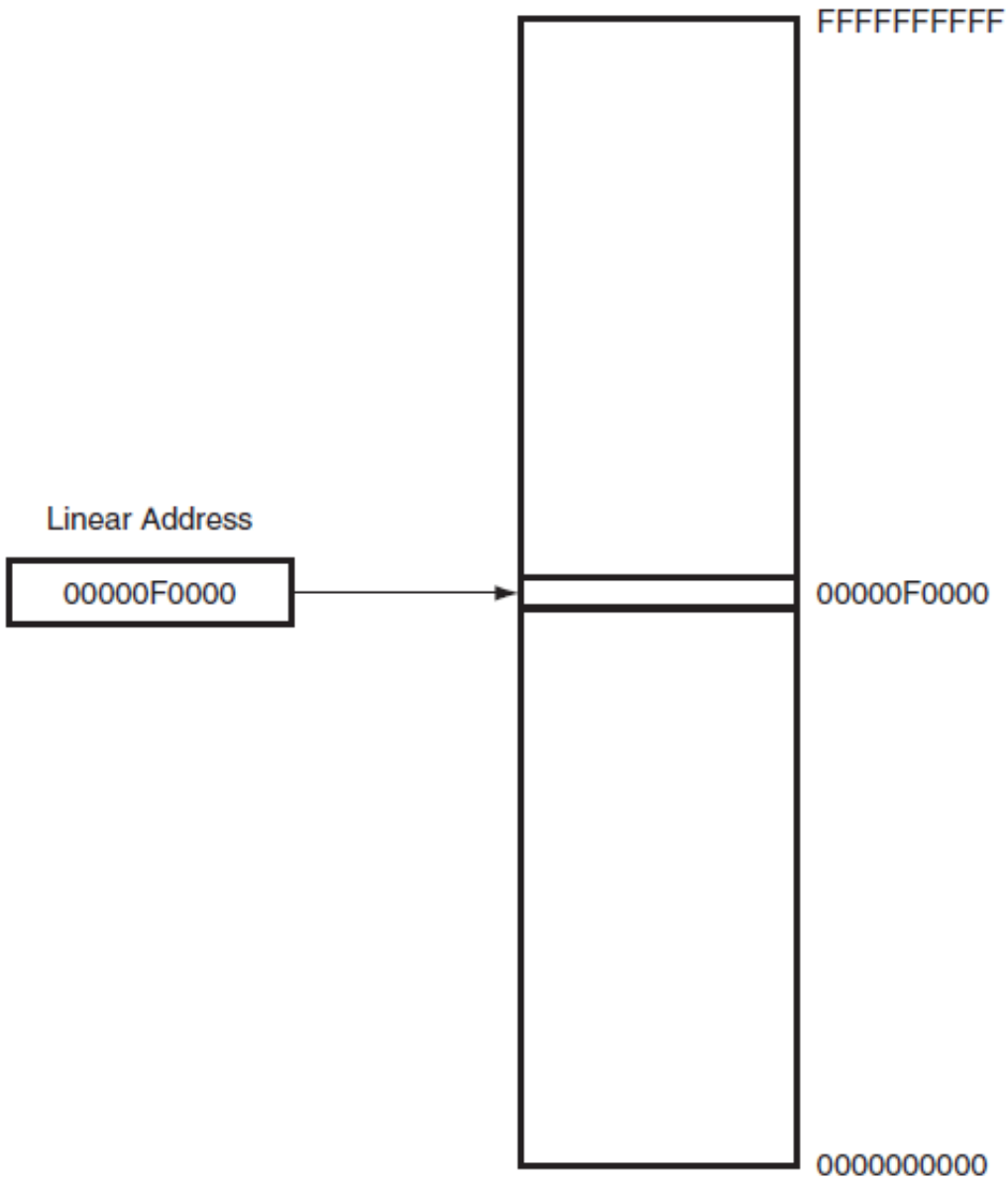
- The flat model does not select the memory address of a segment using the base and limit in the descriptor

- The offset address is the actual physical address in 64-bit mode.

- The 64-bit descriptor has no limit or base address.
- It only contains an access rights byte and the control bits.
- The real mode system is *not available if the processor operates in the 64-bit mode.*
- *Protection and paging* are allowed in the 64-bit mode.
- The CS register is still used in the protected mode operation in the 64-bit mode.

- The flat mode memory contains 1T byte of memory using a 40-bit address.

FIGURE 2–15 The 64-bit flat mode memory model.



Find start and end address of segment in real addressing mode:

- A real mode segment of memory is 64K in length
- once the beginning address is known, the ending address is found by adding **FFFFH.**

- e.g. if a segment register contains 3000H
- then first address of the segment is 30000H(convert to 20 bit),
- And the last address is = 30000+FFFF = 3FFFFH.

In the real mode, Show the starting and ending addresses of each segment located by the following segment register values.
A) 2345H B) ABCDH C) 8899H

•

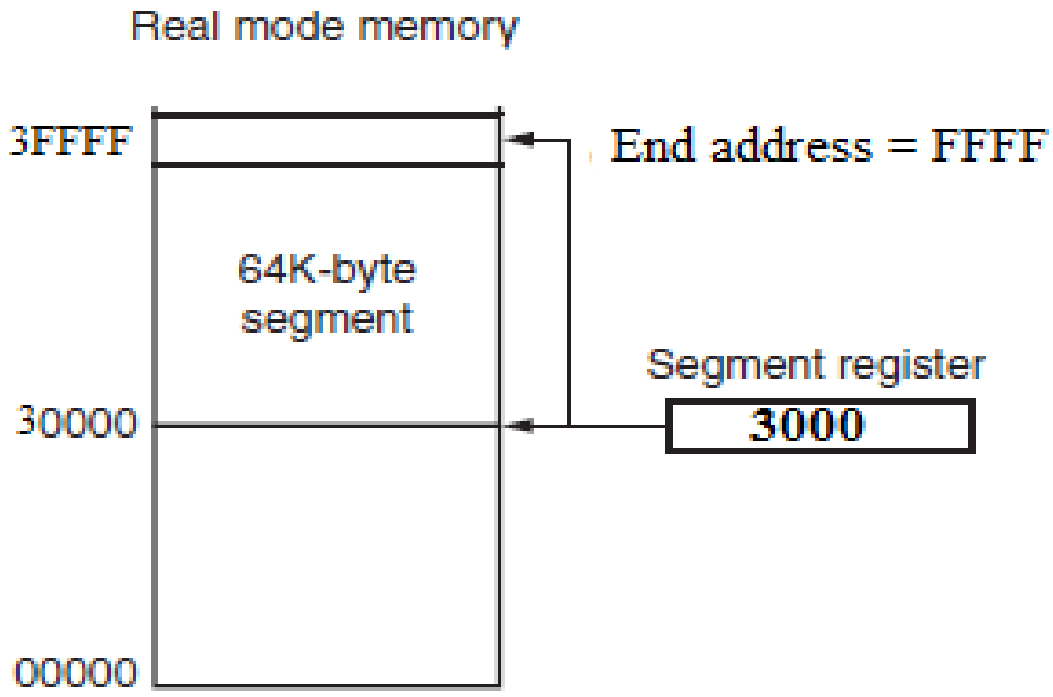


FIGURE The real mode memory-addressing scheme, using a segment address plus an offset.

- 23450+FFFF=3344F
- Here starting address =23450
- And End address = 3344F

8086 Addressing Modes:

- Ways of locating data or operands.
- Describes the type of operands and the way they are accessed for executing an instruction.

8086 data addressing modes:

- **Register addressing:**
- **Immediate addressing:**
- **Direct addressing**
- **Register indirect addressing**
- **Register relative addressing**
- **Base plus index addressing (based indexed)**
- **Base relative plus index addressing (relative based indexed)**
- **Scaled index addressing**

8086 data addressing modes:

- **Register addressing:**
  - Register addressing transfers a copy of a byte or word from the source register or contents of a memory location to the destination register or memory location.
  - Example: MOV CX, DX
    - This instruction copies the word-contents of register DX into register CX.

- **Immediate addressing:**
  - Immediate addressing transfers the source, an immediate byte, word, doubleword, or quadword of data, into the destination register or memory location.
  - Example: The MOV AL, 22H
    - This instruction copies a byte-22H into register AL.

- **Direct Addressing:**
  - Direct addressing moves a byte or word between a memory location and a register.
  - The instruction set does not support a memory-to memory transfer.
  - Example: The MOV CX, LIST
  - This instruction copies the word contents of memory location LIST into register CX.

  - MOV AX, [5000H],

- **Register indirect addressing:**
  - Register indirect addressing transfers a byte or word between a register and a memory location addressed by an index or base register.
  - The index and base registers are BP, BX, DI, and SI.
  - Example: MOV AX, [BX]
    - This instruction copies the word-data from the data segment offset address indexed by BX into register AX.

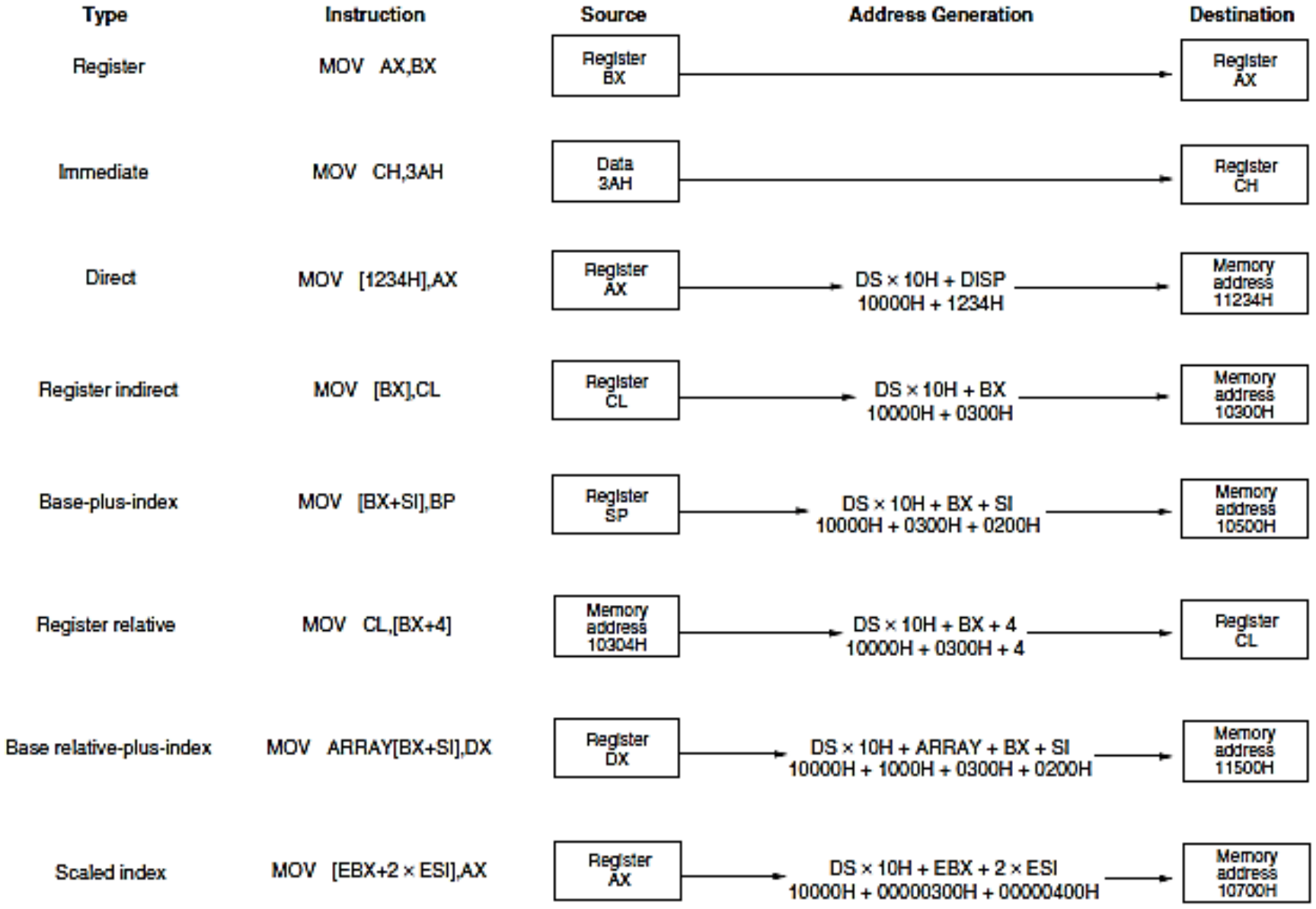
- **Register relative addressing:**
- Register relative addressing moves a byte or word between a register and the memory location addressed by an index or base register plus a displacement.
  - Example: MOV AX,[BX+4]
    - This instruction loads AX from the data segment address formed by BX plus 4.
  - MOV AX,ARRAY[BX].
    - This instruction loads AX from the data segment memory location in ARRAY plus the contents of BX.

- **Base plus index addressing (based indexed)**
- Base-plus-index addressing transfers a byte or word between a register and the memory location addressed by a base register (BP or BX) plus an index register (DI or SI).
  - Example: The MOV [BX+DI], CL
  - This instruction copies the byte-contents of register CL into the data segment memory location addressed by BX plus DI.

- **Base relative plus index addressing (relative based indexed)**
- Base relative-plus-index addressing transfers a byte or word between a index register and the memory location addressed by a base and an index register plus a displacement.
- Example: MOV AX, ARRAY[BX+DI]
  - This instruction uses an address formed by adding ARRAY, BX, and DI
  - MOV AX, [BX+DI+4].
    - This instruction uses an address formed by adding ARRAY, BX, and DI , and 4

- **Scaled index addressing:**
- Scaled-index addressing is available only in the 80386 through the Pentium 4 microprocessor.
- The second register of a pair of registers is modified by the scale factor of 2×, 4×, or 8× to generate the operand memory address.
  - Example: A MOV EDX, [EAX+4*EBX]
  - This instruction loads EDX from the data segment memory location addressed by EAX plus four times EBX.
  - Scaling allows access to word (2x), doubleword (4x), or quadword (8x) memory array data.

- **RIP relative addressing:**
  - This addressing mode is only available to the 64-bit extensions on the Pentium 4 or Core2.
  - This mode allows access to any location in the memory system by adding a 32-bit displacement to the 64-bit contents of the 64-bit instruction pointer.
  - Example,
    - if RIP = 1000000000H and a 32-bit displacement is 300H, the location accessed is 1000000300H.



Notes: EBX = 00000300H, ESI = 00000200H, ARRAY = 1000H, and DS = 1000H

FIGURE 8086–Core2 data-addressing modes.

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.

Features of 80386

- 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 with 80286.

Features of 80386

- 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 organize 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.

Paging

- Paging operation:
- Memory management technique used for physical memory multitasking system.

Comparison-segmentation and Paging

Paging	segmentation
Paging divides Physical memory into fixed size pages.	Physical memory is divided into variable size segments
Pages do not have any logical relation with program.	Segments are supposed to be the logical segments of the program
The user specified address is divided by CPU into a page number and offset.	The user specifies each address by two quantities a segment number and the offset (Segment limit).
The hardware decides the page size.	The segment size is specified by the user.
Paging involves a page table that contains base address of each page.	Segmentation involves the segment table that contains segment number and offset (segment length).

- **Paging Unit :**
  - Paging unit converts linear address provided by segmentation unit into physical address.
  - Paging unit translates complete map of task into pages, each of size 4K and each task is handled in terms of pages.
  - Paging unit handles every task in terms of 3 components
    - Page directory
    - Page table
    - Page itself

- **Page Directory:**
- This is at most 4Kbytes in size
- Each directory entry is of 4 bytes, total 1024 entries in a directory
- Upper 10 bits of the linear address are used as an index to the corresponding page directory entry.
- Page directory entries points to the page table.

- **Page Table:**
  - Each table is of 4Kbyte in size and contain maximum 1024 entries.
  - Page table entries contain the starting address of the page and statistical information about the page.
  - Upper 20 bits of the page frame address is combined with lower 12 bit linear address.
  - The address bits A_{12}-A_{21} (10 bits)are used to select 1024 page table entries.
  - Page table can be shared between the tasks.

- **Page descriptor base register:**
  - The control register CR_2 is used to store 32 bit linear address at which the previous page fault was detected.
  - The physical starting address of the current page directory is stored in the CPU register **CR_3,** also called the page directory base register (PDBR).
  - Lower 12 bits of CR_3 are always 0 to ensure the page size aligned with the directory.

- **Page Frame Address**
  - Is the physical starting address of a page.
  - Because pages are located on 4K boundaries, the low-order 12 bits are always zero.
  - In a page directory, the page frame address is the address of a page table.

- In paging, the **page table** maps the **logical address to the physical address**, and it contains base address of each page stored in the frames of physical memory space.

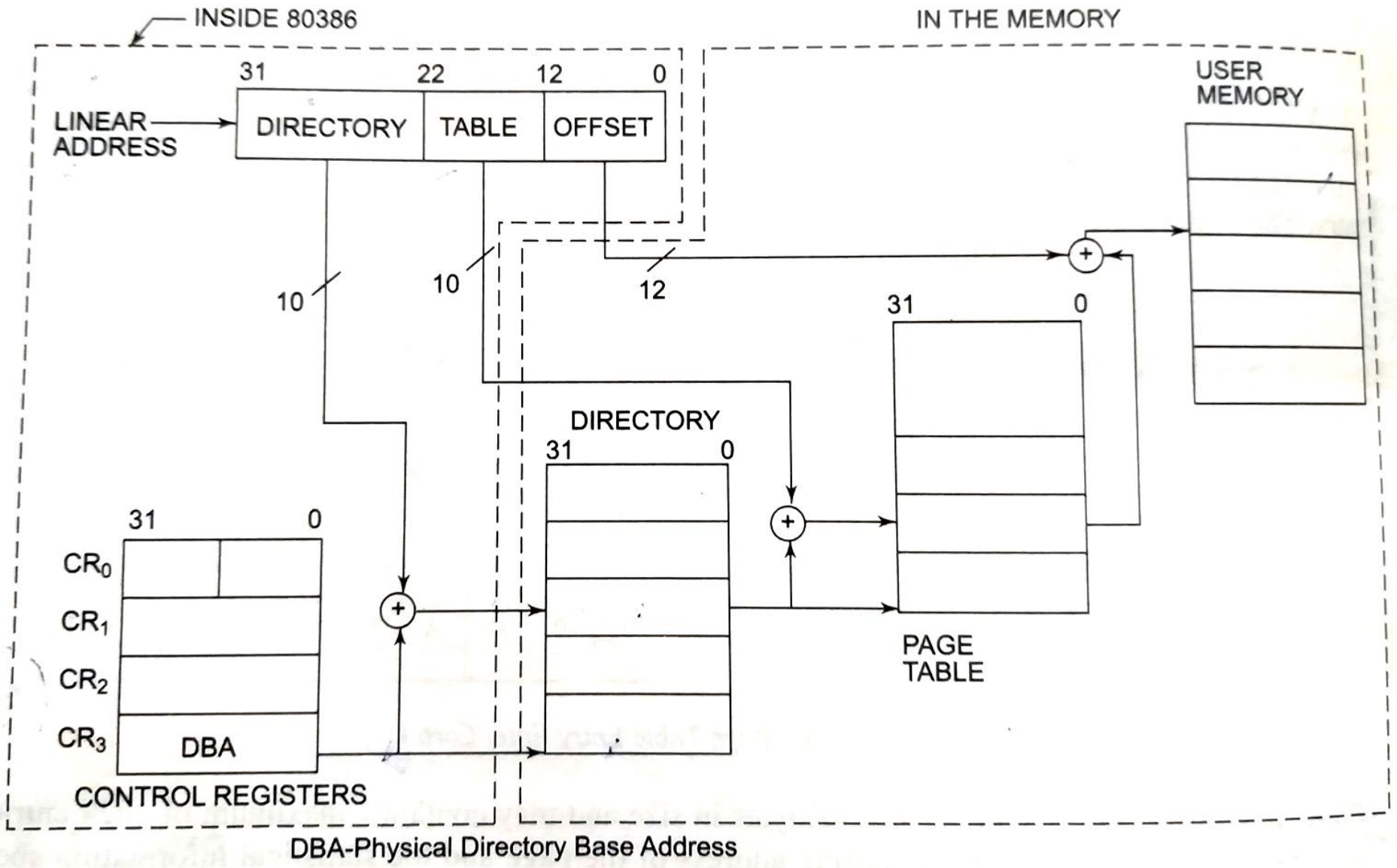


Fig. 10.8 *Paging Mechanism of 80386 (Intel Corp.)*

Thank you