Computer Engineering II

Electrical Engineering Department

Third Stage

2nd Semester 2024-2025

Instructor: Asst. Lecturer

Adnan Ali Abdullah

THE 80386 MICROPROCESSOR

The 80386 microprocessor is a full 32-bit version of the earlier 8086/80286.

The 80386 microprocessor features

- The 80386DX addresses **4G bytes of memory** through its **32-bit data bus** and **32-bit address**.
- Multitasking,
- Memory management,
- Virtual memory (with or without paging),
- Software protection,
- A large memory system.
- 32 bit register size(EAX,ABX,ESI…..)
- Operating frequency (10MHz-40MHz)

The 80386 can switch between protected mode and **real mode without resetting the microprocessor**. Switching from protected mode to real mode was a problem on the 80286 microprocessor because it required a hardware reset.

- Two versions of the 80386 are commonly available: the **80386DX**, and the **80386SX**
- The 80386DX, which is illustrated and described in this lecture.
- Figure 1 illustrates the pin-out of the 80386DX microprocessor. The 80386DX is packaged in a 132-pin PGA (pin grid array).

The function of each 80386DX group of pins follows:

A_{31}–A_2 — **Address bus** connections address any of the 1G × 32 (4G bytes) memory locations found in the 80386 memory system. Note that A_0 and A_1 are encoded in the bus enable ($\overline{BE3}$–$\overline{BE0}$) to select any or all of the four bytes in a 32-bit-wide memory location.

D_{31}–D_0 — **Data bus** connections transfer data between the microprocessor and its memory and I/O system. Note that the 80386SX contains D_{15}–D_0.

$\overline{BE3}$–$\overline{BE0}$ — **Bank enable** signals select the access of a byte, word, or doubleword of data. These signals are generated internally by the microprocessor from address bits A_1 and A_0. On the 80386SX, these pins are replaced by $\overline{BHE}$, $\overline{BLE}$, and A_1.

M/$\overline{IO}$ — **Memory/IO** selects a memory device when a logic 1 or an I/O device when a logic 0. During the I/O operation, the address bus contains a 16-bit I/O address on address connections A_{15}–A_2.

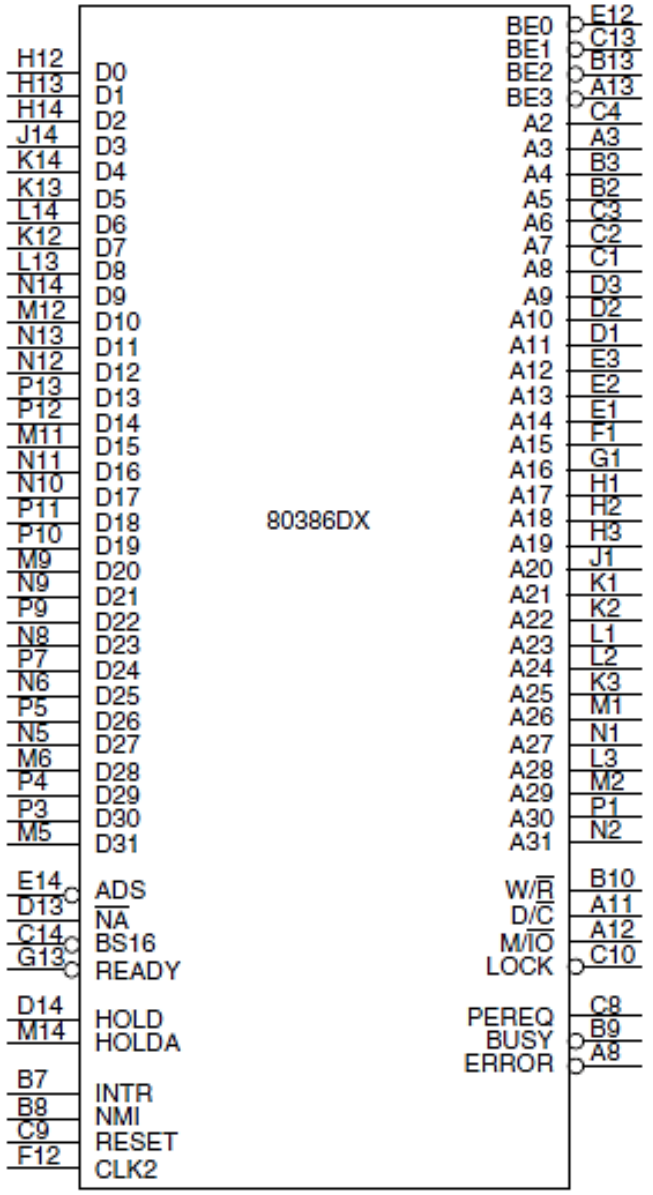

FIGURE 1 The pin-outs of the 80386DX

W/$\overline{R}$	**Write/$\overline{Read}$** indicates that the current bus cycle is a write when a logic 1 or a read when a logic 0.
$\overline{ADS}$	The **address data strobe** becomes active whenever the 80386 has issued a valid memory or I/O address. This signal is combined with the W/$\overline{R}$ signal to generate the separate read and write signals present in the earlier 8086–80286 microprocessor-based systems.
RESET	**Reset** initializes the 80386, causing it to begin executing software at memory location FFFFFFF0H. The 80386 is reset to the real mode, and the leftmost 12 address connections remain logic 1s (FFFH) until a far jump or far call is executed. This allows compatibility with earlier microprocessors.
CLK_2	**Clock times 2** is driven by a clock signal that is twice the operating frequency of the 80386. For example, to operate the 80386 at 16 MHz, apply a 32 MHz clock to this pin.
$\overline{READY}$	**Ready** controls the number of wait states inserted into the timing to lengthen memory accesses.
$\overline{LOCK}$	**Lock** becomes a logic 0 whenever an instruction is prefixed with the LOCK: prefix. This is used most often during DMA accesses.
D/$\overline{C}$	**Data/control** indicates that the data bus contains data for or from memory or I/O when a logic 1. If D/$\overline{C}$ is a logic 0, the microprocessor is halted or executes an interrupt acknowledge.

$\overline{\text{BS16}}$	**Bus size 16** selects either a 32-bit data bus ($\overline{\text{BS16}} = 1$) or a 16-bit data bus ($\overline{\text{BS16}} = 0$). In most cases, if an 80386DX is operated on a 16-bit data bus, we use the 80386SX that has a 16-bit data bus. On the 80386EX, the $\overline{\text{BS8}}$ pin selects an 8-bit data bus.
$\overline{\text{NA}}$	**Next address** causes the 80386 to output the address of the next instruction or data in the current bus cycle. This pin is often used for pipelining the address.
HOLD	**Hold** requests a DMA action.
HLDA	**Hold acknowledge** indicates that the 80386 is currently in a hold condition.
$\overline{\text{PEREQ}}$	The **coprocessor request** asks the 80386 to relinquish control and is a direct connection to the 80387 arithmetic coprocessor.
$\overline{\text{BUSY}}$	**Busy** is an input used by the WAIT or FWAIT instruction that waits for the coprocessor to become not busy. This is also a direct connection to the 80387 from the 80386.
$\overline{\text{ERROR}}$	**Error** indicates to the microprocessor that an error is detected by the coprocessor.
INTR	An **interrupt request** is used by external circuitry to request an interrupt.
NMI	A **non-maskable interrupt** requests a non-maskable interrupt as it did on the earlier versions of the microprocessor.

The Memory System

The physical memory system of the 80386DX is 4G bytes in size and is addressed as such. If virtual addressing is used, 64T bytes are mapped into the 4G bytes of physical space by the memory management unit and descriptors. (Note that virtual addressing allows a program to be larger than 4G bytes if a method of swapping with a large hard disk drive exists.) Figure–2 shows the organization of the 80386DX physical memory system.

The memory is divided into four 8-bit wide memory banks, each containing up to 1G bytes of memory.

This 32-bit-wide memory organization allows bytes, words, or double words of memory data to be accessed directly.

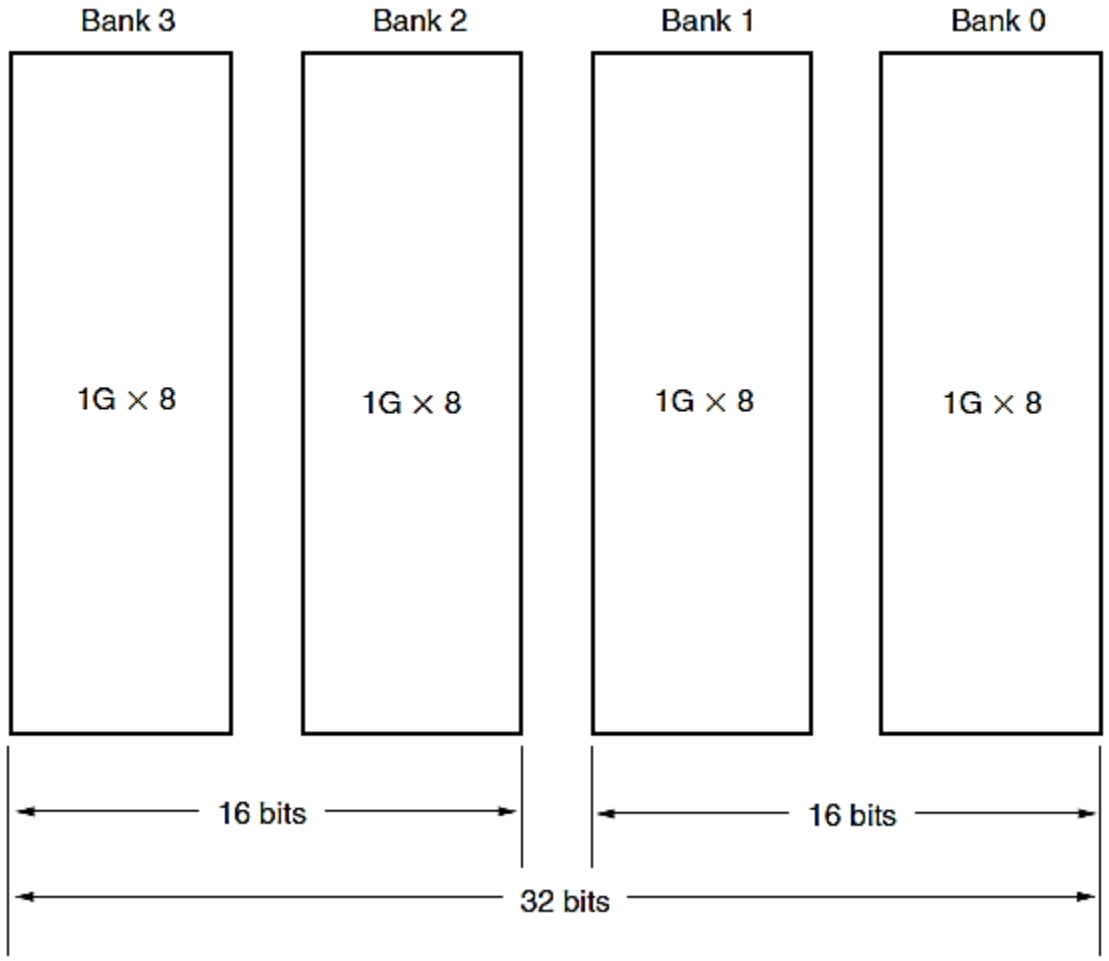

FIGURE 2 The memory system for the 80386 microprocessor.

The 80386DX transfers up to a 32-bit-wide number in a single memory cycle, whereas the early 8088 requires four cycles to accomplish the same transfer, and the 80286 and 80386SX require two cycles.

Each memory byte is numbered in hexadecimal as they were in prior versions of the family.
The difference is that the 80386DX uses a 32-bit-wide memory address, with memory bytes numbered from location 00000000H to FFFFFFFFH.

The two memory banks in the 8086, 80286, and 80386SX system are accessed via $\overline{BLE}$ (A0 on the 8086 and 80286) and $\overline{BHE}$ In the 80386DX, the memory banks are accessed via four bank enable signals, $\overline{BE3} - \overline{BE0}$ This arrangement allows a single byte to be accessed when one bank enable signal is activated by the microprocessor. It also allows a word to be addressed when two bank enable signals are activated.

Memory location 00000000H is in bank 0, location 00000001H is in bank 1, location 00000002H is in bank 2, and location 00000003H is in bank 3

The 80386DX does not contain address connections A_0 and A_1 because these have been encoded as the bank enable signals.

Buffered System

- Needed to increase fan out
- Used to protect Processor from unexpected signals
- 74f244 try-state buffers for address bus and control signals, and 74f245 by-directional try-state buffers for data bus
- The HLDA signal is used to enable all buffers in a system that uses direct memory access.

Pipelines and Caches

The cache memory is a buffer that allows the 80386 to function more efficiently with lower DRAM speeds.

A pipeline is a special way of handling memory accesses so the memory has additional time to access data.

Some technique must be found to interface these memory devices, which are slower than required by the microprocessor

Three techniques are available:
interleaved memory, **caching**, and a **pipeline**.

1. Interleaved Memory

✅ **Concept:**

Memory is divided into **multiple "banks"**, and data access is alternated between these banks.

This allows the CPU to **access memory faster**, since while one bank is being accessed, another can be prepared.

💡 **How It Works:**

•Suppose we have Bank 0, Bank 1, Bank 2, Bank 3.

•The CPU reads data from Bank 0.

•While it's processing that, the system fetches the next data from Bank 1, and so on.

📈 **Benefit:**

•Reduces **memory access time**.

•Makes memory appear faster without changing the physical memory.

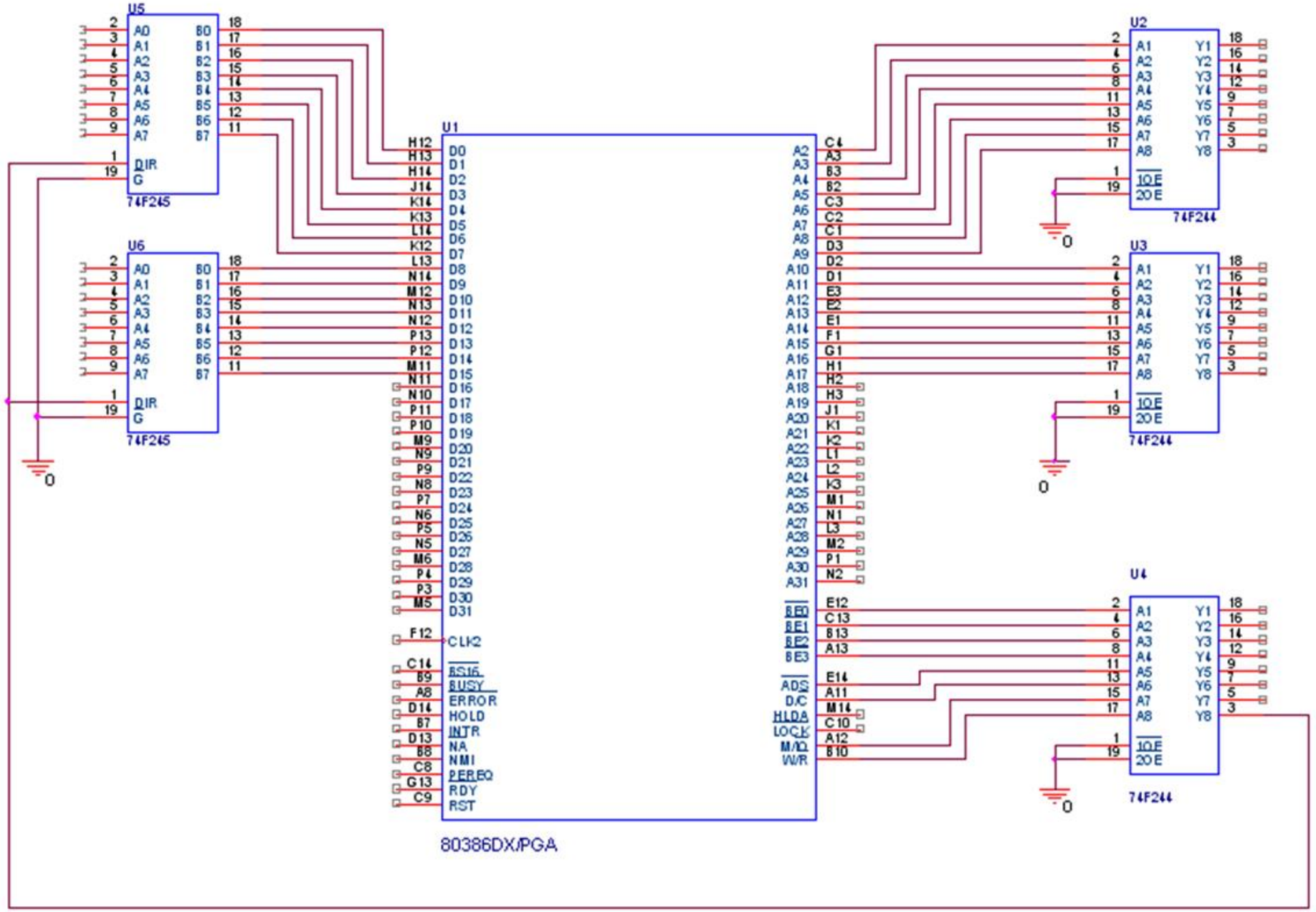

FIGURE 3 A fully buffered 25 MHz 80386DX.

2. Caching

✅ **Concept:**

Use a **very fast, small memory** called a **cache** to store data that the CPU is likely to need again.

💡 **How It Works:**

•When the CPU needs data, it first checks the cache.

•If the data is found (called a **cache hit**), it's accessed instantly.

•If not (a **cache miss**), the data is fetched from slower main memory and stored in the cache.

📈 **Benefit:**

•**Reduces wait time** for frequently used data.

•Greatly **increases system performance**.

3. Pipeline

✅ **Concept:**

The CPU breaks down instruction execution into **multiple stages**, and executes different parts of multiple instructions **in parallel**—like an assembly line.

💡 **How It Works:**

A typical pipeline may have:

1.Fetch instruction

2.Decode it

3.Execute it

4.Access memory

5.Write back result

So while instruction 1 is in stage 3, instruction 2 is in stage 2, and instruction 3 is in stage 1 — **all at once**.

The Input/output System

The 80386 input/output system is the same as that found in any Intel 8086 family microprocessor based system.
There are 64K different bytes of I/O space available if isolated I/O is implemented.

Type	Description
Isolated I/O	Uses special IN/OUT instructions. 64K I/O space. Separate from memory.
Memory-mapped I/O	I/O devices treated as memory. Any instruction can access I/O. 4GB space.

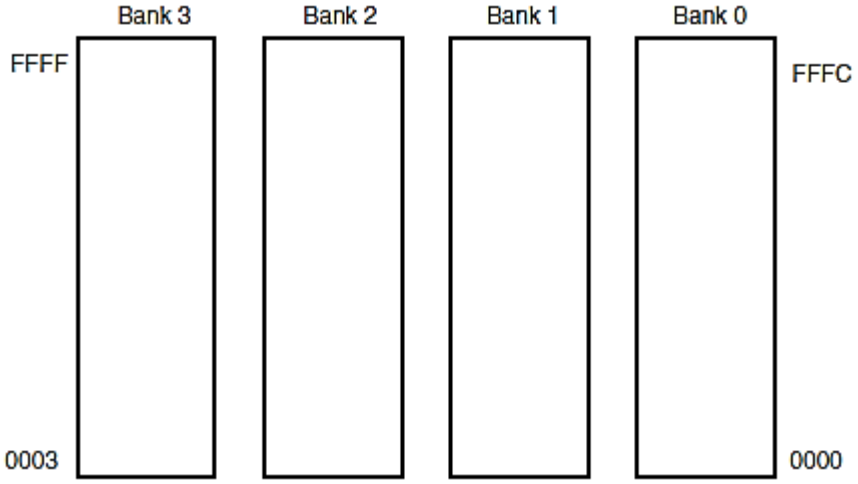

FIGURE 4 The isolated I/O map for the 80386

2. I/O Addressing:

- Uses **A_2–A_{15}** for port addressing.
- **BE3–BE0** selects byte/word/double word.
- Range: **0000H to FFFFH** (64K bytes for isolated I/O).
- Devices may overlap memory addresses (e.g., coprocessors).

3. I/O Protection (80386):

- Protected mode blocks I/O access via privilege levels.
- Blocked I/O = **interrupt (type 13)**.
- Controlled via **Task State Segment (TSS)**.

Memory and I/0 Control Signals

The memory and I/O are controlled with separate signals.

The M/$\overline{IO}$ signal indicates whether the data transfer is between the microprocessor and the memory (M/$\overline{IO}$=1) or I/O (M/$\overline{IO}$=0).

In addition to M/$\overline{IO}$, the memory and I/O systems must read or write data.

The W/$\bar{R}$ signal is a logic 0 for a read operation and a logic 1 for a write operation.

SPECIAL 80386 REGISTERS

- Control registers CR_0–CR_3
- Facilitate debugging DR_0–DR_7
- Test paging TR_6-TR_7

Control Registers (CR0-CR3)
Purpose: These registers control system-wide functions such as paged memory management and protected mode operation.
CR0: Contains system control flags that indicate conditions like paging and protected mode status.
CR3: Used to locate the page table directory when paging is enabled.
Debug Registers (DR0-DR7)
Purpose: Provide advanced debugging capabilities, including data and instruction breakpoints.
DR0-DR3: Store addresses for breakpoints.
DR7: Controls debug conditions and enables/disables breakpoints.
Test Registers (TR6-TR7)
Purpose: Used for self-testing, specifically TLB testing in the 80386.
TR6: Test command register.
TR7: Test data register.
Note: The term “Test paging registers” is not standard; the test registers are not directly related to paging but are used for testing purposes.

MEMORY MANAGEMENT

The memory-management unit (MMU) within the 80386 is similar to the MMU inside the 80286, except that the 80386 contains a paging unit not found in the 80286.

The MMU performs the task of converting linear addresses, as they appear as outputs from a program, into physical addresses that access a physical memory location located anywhere within the memory system.

The 80386 uses the paging mechanism to allocate any physical address to any logical address.

Even though a program accesses memory using a specific logical address (e.g., A0000h), the actual physical location in RAM may be completely different (e.g., 100000h). This is possible because the Intel 80386 processor supports paging, a memory management technique that translates linear addresses (generated by the program) into physical addresses in RAM.

The program remains unaware of this translation—it just sees the address it expects (A0000h), while the CPU silently redirects it to the correct physical location (100000h).

THE MEMORY PAGING MECHANISM

The paging mechanism allows any linear (logical) address, as it is generated by a program, to be placed into any physical memory page, as generated by the paging mechanism. A linear memory page is a page that is addressed with a selector and an offset in either the real or protected mode.

A physical memory page is a page that exists at some actual physical memory location. For example, linear memory location 20000H could be mapped into physical memory location 30000H, or any other location, with the paging unit. This means that an instruction that accesses location 20000H actually accesses location 30000H.

Each 80386 memory page is 4K bytes long. Paging allows the system software to be placed at any physical address with the paging mechanism. Three components are used in page address translation: the page directory, the page table, and the actual physical memory page. Note that EEM386.EXE, the extended memory manager, uses the paging mechanism to simulate expanded memory in extended memory and to generate upper memory blocks between system ROMs.

Why Paging is Useful?

- Allows **virtual memory** (disk swapping when RAM is full).
- Enables **memory protection** (prevents programs from accessing each other's memory).
- Supports **multitasking** (each program thinks it has full memory access).

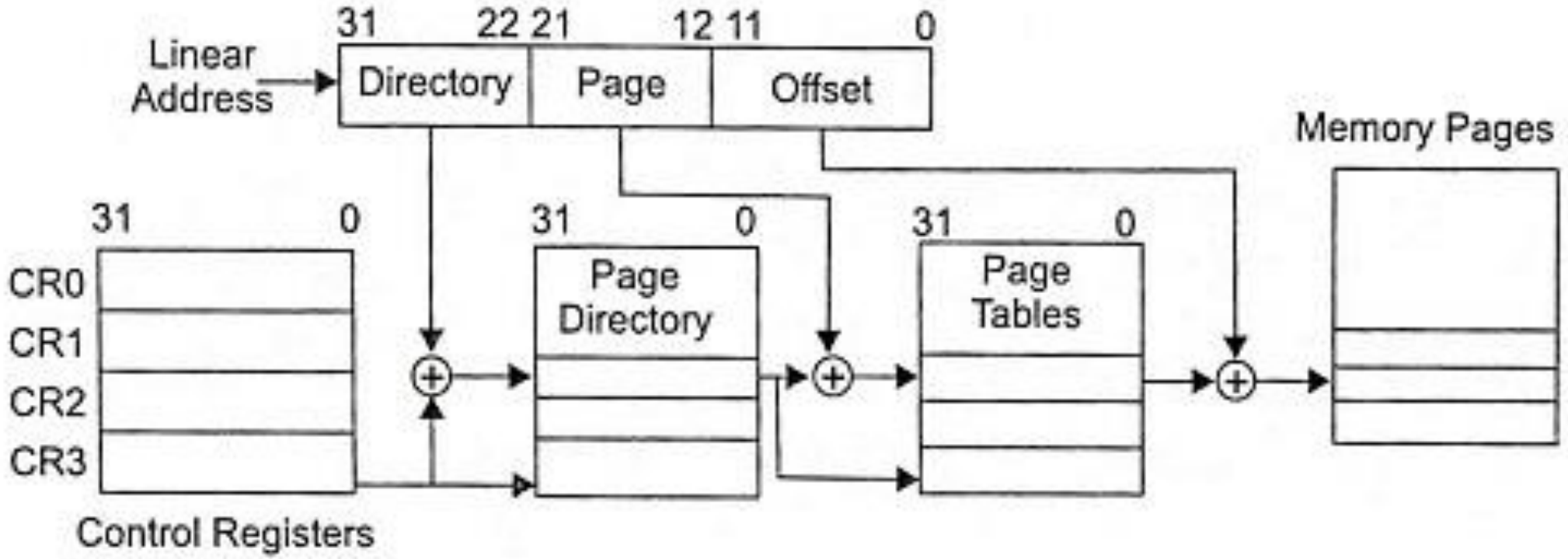

Fig. 11.41 *Paging technique of 80386*