Dec Hex Bin
21 15 00010101

ORG ; TWENTY-ONE

386 MICROPROCESSOR:REAL vs. PROTECTED MODE

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

The x86 PC
assembly language, design, and interfacing
fifth edition

Prentice Hall

OBJECTIVES

this chapter enables the student to:

- List the additional features implemented on the 80186, 80286, and 80386.
- State the purpose of designing two modes, real and protected, into the 80386.
- Code Assembly language instructions using the new scaled index addressing mode of the 386.
- Code Assembly language instructions using the new instructions of the 386.
- State the purpose of each pin of the 80386 microprocessor.

OBJECTIVES (*cont*)

this chapter enables the student to:

- Describe the data misalignment problem with 386 programs and discuss how to resolve the problem.
- Describe how protection of user & system programs is accomplished in the 386 in protected mode.
- Contrast and compare the two methods of virtual memory implementation:
  - Paging and segmentation.
- Describe the methods of converting from logical to physical addresses in 386 protected mode.

21.1: 80386 IN REAL MODE
evolution of x86

- Two dominant trends in x86 system software:
  - 1. Software that runs on any x86, including 8088/86.
  - 2. 32-bit, 386 based software for 386 & higher machines.

21.1: 80386 IN REAL MODE
evolution of x86 – 80186

- Intel did a survey & found many use the 8088/86 with other peripheral chips, such as 8237 DMA controller, 8254 timer & 8259 interrupt controller.
  - Putting a portion of these chips with the 8088/86 on a single chip led to the 68-pin 80186/88.
    - Very successful in the embedded controller market.
- 80186/88 is a 68-pin chip that includes
  - Clock generator; Two 20-bit DMA channels.
  - Three 16-bit programmable counters; Interrupt controller.
  - Programmable wait-state generator & chip select decoder.
  - 80186 has a 16-bit data bus; 80188 an 8-bit bus.
    - The data bus is multiplexed with the address bus.

21.1: 80386 IN REAL MODE
evolution of x86 - 80286

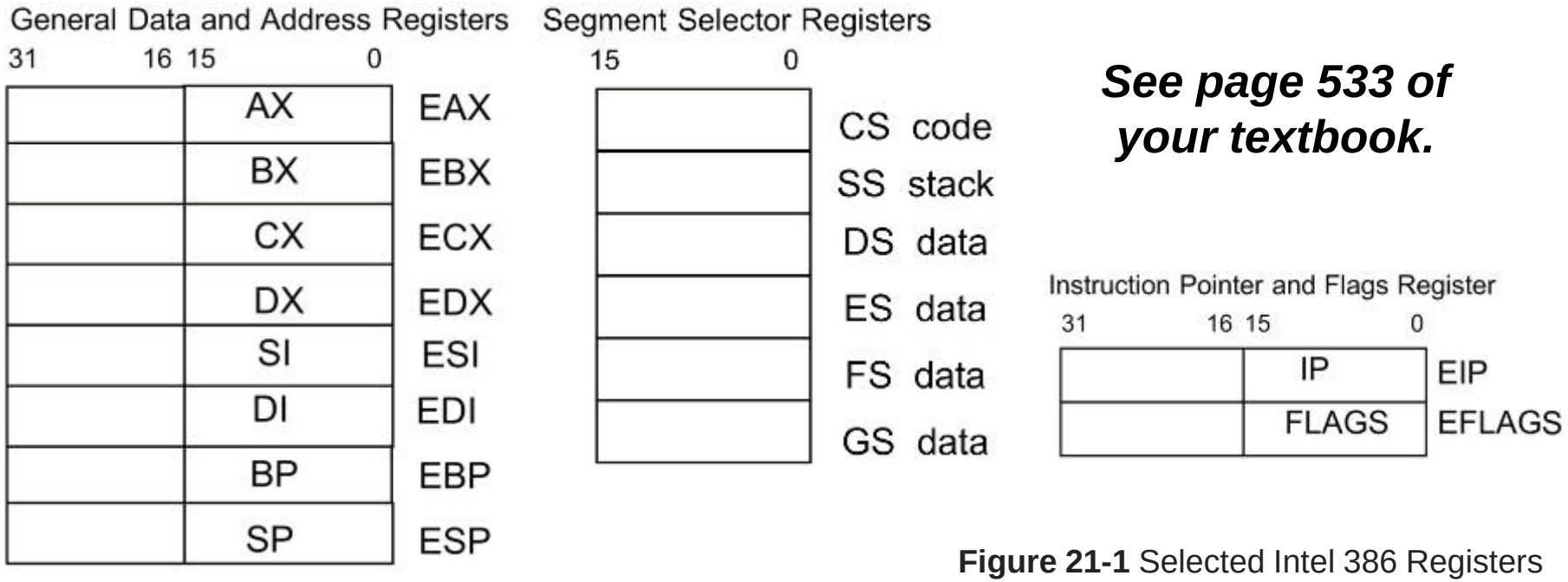
- Demand for a more powerful CPU led Intel to use more than 100,000 transistors on the new PGA (pin grid array) packaged 80286.
  - Downwardly compatible with 80186/88 processor.
- 80286 was a major improvement over the core 8086:
  - Separate pins for address/data buses.
    - No need for demultiplexing the buses.
  - Memory cycle time was reduced from 4 clocks to 2 clocks.
  - Introduction of virtual memory.
  - Two different modes: real mode and protected mode.

21.1: 80386 IN REAL MODE
evolution of x86 - 80386

- While downwardly compatible, there are major changes in the 32-bit 80386 architecture:
  - Data bus was increased from 16 bits to 32 bits.
    - Internally and externally.
  - All registers were extended to 32 bits.
  - Address bus was increased to 32 bits.
    - 4 gigabytes (2^{32}) of physical memory addressing capability.
  - Paging virtual memory mechanism was introduced.
  - A new addressing mode called scaled index was added.
  - Many new bit-manipulation instructions were added.
    - For both real mode and protected mode.
  - 80386 can switch from protected to real mode by software.

21.1: 80386 IN REAL MODE
32-bit registers

- In the 80386, register sizes were extended from 16 to 32 bits & register names changed to reflect this.
  - EAX is the extended AX, EBX the extended BX, etc.
    - To access 32-bit registers, letter E must be in the code.



See page 533 of your textbook.

Figure 21-1 Selected Intel 386 Registers

21.1: 80386 IN REAL MODE
32-bit registers

- General-purpose registers AX, BX, CX, and DX are accessible in 8086, as well as extended, format.
  - EAX is accessible as AL, AH, AX, and EAX.
    - The upper 16-bit part is not accessible as a separate register.

Example 21-3

Load EAX with 7698E35FH and move it among the 8-, 16-, and 32-bit registers of the 386.

Solution:

```
MOV   EAX,7698E35FH     ;EAX=7698E35F (AX=E35F,AH=E3,AL=5F)
MOV   EDX,EAX           ;EDX=EAX=7698E35F
MOV   CH,AL             ;CH=AL=5F
MOV   DI,AX             ;DI=AX=E35F
MOV   ESI,EDX           ;ESI=EDX=7698E35F
ROR   EAX,16            ;rotate EAX right 16 times (EAX=E35F7698H)
MOV   BX,AX             ;BX=AX=7698H
MOV   CL,AL             ;CL=AL=98H
```

21.1: 80386 IN REAL MODE
which end goes first?

- In storing data, 386 places the least significant byte (little end of the data) in the low address.
  - 9FH goes to low address 8000.

Example 21-4

Show how data is placed after execution of the following code.

```
MOV   EAX,7698E39FH     ;EAX=7698E39F
MOV   [4524],AX
MOV   [8000],EAX
```

Solution: For "MOV [4524],AX" we have

```
DS:4524 =(9F)
DS:4525 =(E3)
```

and for "MOV [8000],EAX" we have

```
DS:8000 = (9F)
DS:8001 = (E3)
DS:8002 = (98)
DS:8003 = (76)
```

Little Endian

21.1: 80386 IN REAL MODE
general registers as pointers

- Another major change in 80386 is use of general registers such as EAX, ECX, and EDX as pointers.

Table 21-1: Addressing Modes for the 80386

Addressing Mode	Operand	Default Segment
Register	Register	None
Immediate	Data	None
Direct	[OFFSET]	DS
Register indirect	[BX]	DS
	[SI]	DS
	[DI]	DS
	[EAX]	DS
	[EBX]	DS
	[ECX]	DS
	[EDX]	DS
	[ESI]	DS

All 32-bit general-purpose registers can be used for pointers into data segments.

386 also allows use of displacement for 32-bit register pointers.

See the table of addressing modes on page 535 of your textbook.

21.1: 80386 IN REAL MODE
scaled index addressing mode

- Scaled index addressing mode allows access of multidimensional arrays with ease.
  - Any of the 32-bit registers, except ESP, can be used as a pointer, multiplied by a scaled factor of 1, 2, 4, or 8.

Table 21-2: 386 Scaled Index Addressing Mode

Scaled Index	Default Segment
[EAX]	DS
[EBX]	DS
[ECX]	DS
[EDX]	DS
[ESI]	DS
[EDI]	DS
[EBP]	SS
[ESP]	SS

21.1: 80386 IN REAL MODE
scaled index addressing mode

- Scaling (multiplication) factors 1, 2, 4, 8 correspond to byte, word, doubleword & quadword operands.
  - Only 32-bit register pointers can be used for this mode.

Example 21-5

Find the effective address in each of the following cases. Assume that ESI = 200H, ECX = 100H, EBX = 50H, and EDI = 100H.

(a) MOV AX,[2000+ESI*4]
(b) MOV AX,[5000+ECX*2]
(c) MOV ECX,[2400+EBX*4]
(d) MOV DX,[100+EDI*8]

Solution:

(a) EA (effective address) is 2000H + 200H × 4 = 2000 + 800H = 2800H. Therefore, the logical address of the operand moved into AX is DS:2800H.
(b) By the same token we have EA = 5000H + 100H × 2 = 5000H + 200 = 5200H.
(c) EA = 2400H + 4 × 50H = 2400H + 140H = 2540H.
(d) 100H + 8 × 100H = 100H + 800H = 900H.

21.1: 80386 IN REAL MODE
scaled index addressing mode

- Example 21-6 shows how the scaled index addressing mode is used.
  - A 16-bit register cannot be used as a scaled index.

Example 21-6

Using the scaled index addressing mode, write an Assembly language program to add 5 operands of 32-bit size and save the result.

Solution:

```
                .MODEL SMALL
                .386
                .STACK 300H
                .DATA
MYDATA          DD     234556H,0F983F5H,6754AE2H,0C5231239H,0AF34ACB4H
RESULT          DQ     ?
                .CODE
                MOV    AX,@DATA
```

For an Assembly program to run under DOS, the effective address should not exceed FFFFH.

See the entire program listing on page 536 of your textbook.

21.1: 80386 IN REAL MODE
new 386 instructions

- New 386 instruction **CDQ** (convert doubleword to quadword) copies the sign bit of EAX, D31, to all the bits of EDX.

21.1: 80386 IN REAL MODE
MOVSX and MOVZX instructions

- Because in all sign-extend instructions, the accumulator sign is extended Intel introduced the MOVSX and MOVZX instructions.
  - In MOVSX, the sign bit of any register (or even a memory location) can be extended (copied) into any register.
    - Used to sign-extend the operand in signed number arithmetic to prevent overflow problems.
  - The MOVZX instruction is used in unsigned arithmetic.
    - Zero-extends the contents of a register or memory location.
    - See Example 21-7 on page 537.

21.1: 80386 IN REAL MODE
bit scan instructions

- 386 has new instructions allowing a program to scan an operand from LSB to MSB or from MSB to LSB, to find the first high bit (= 1).
  - If scanning is done from the least significant bit (D0) toward higher bits, BSF (bit scan forward) is used.
  - If scanning is done from the most significant bit (D31) toward the lower bits, BSR (bit scan reverse) is used.

21.1: 80386 IN REAL MODE
bit scan instructions

- When the first high is found, scanning is stopped & bit position is written into the destination register.
  - Bit position is numbered from D0 (LSB) to D31 (MSB), regardless of the direction of scanning.

Example 21-8

Find the register contents after the execution of the following code.

```
(a)     MOV   BX,4578H
        BSF   DX,BX ;scan BX and put the position of the first high into DX
(b)     MOV   ECX,3A9H
        BSR   EAX,ECX ;scan ECX from D31 down and put position of
                      ;first high into EAX
```

Solution:

(a) DX = 03 since scanning 4578H = 0100 0101 0111 1000B from right to left yields 1 in D3.
(b) EAX = 9 since scanning 000003A9H = 0000 0000 0000 0000 0000 0011 1010 1001 from D31 toward D0 yields the first high in D9; therefore, EAX = 9.

21.2: 80386: A HARDWARE VIEW

- The Intel 80386 has a 32-bit external data bus.
  - 80386SX refers to the 386 with a 16-bit external data bus.

Table 21-3: Data Bus Selection/BE

Data Bus	Byte Enable
D7 – D0	BE0
D15 – D8	BE1
D23 – D16	BE2
D31 – D24	BE3

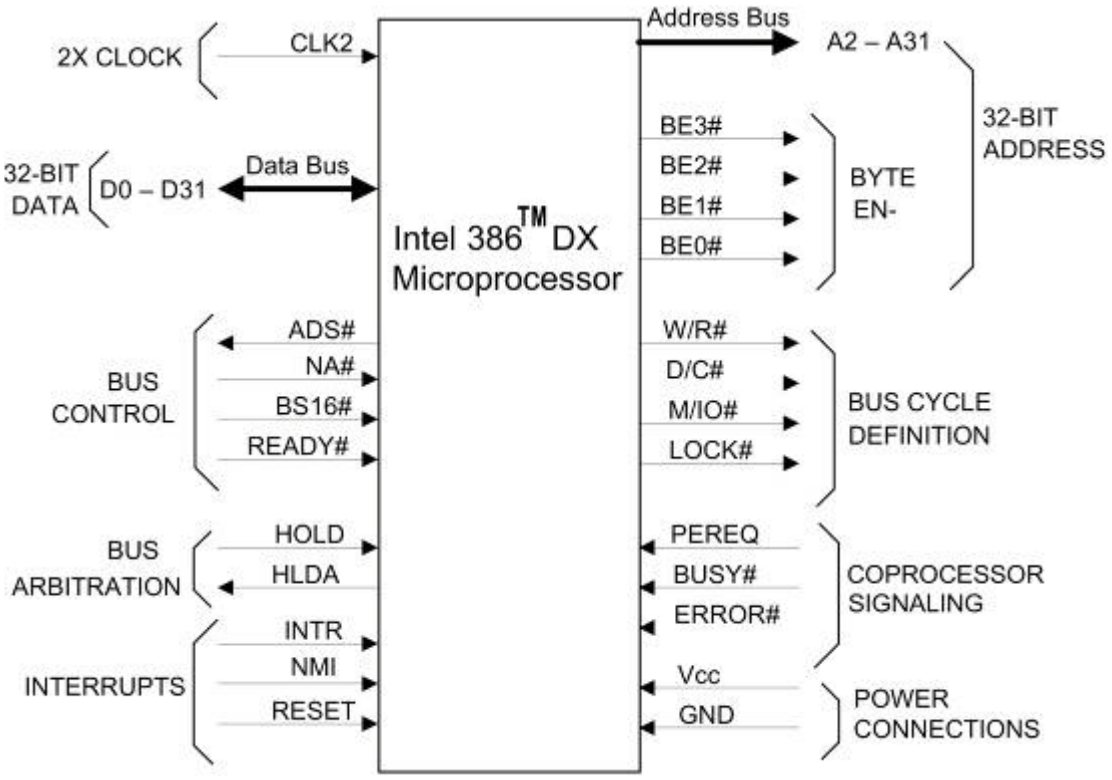


Fig. 21-2 80386 Block Diagram (# indicates *active-low*)

21.2: 80386: A HARDWARE VIEW
overview of pin functions

- **D31–D0** (data bus)
  - Provide the 32-bit data path to the system board.
  - 8-bit data chunks: D0–D7; D8–D15. D16–D23; D24–D31.
  - Each 8-bit data bus is accessed by a separate byte enable pin (**BE**).

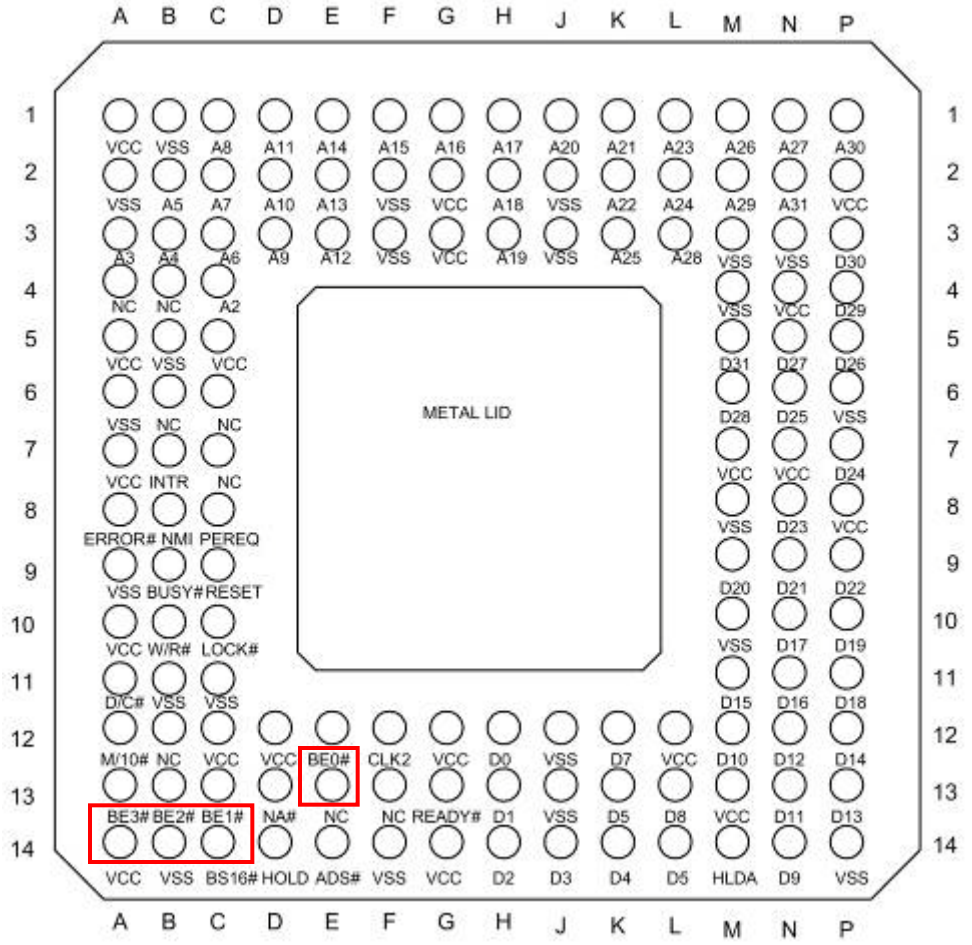


Fig. 21-3 80386 PGA Pins

21.2: 80386: A HARDWARE VIEW
overview of pin functions

- **A31–A2** and **BE0, BE1, BE2, BE3** - provide the 32-bit address path to the system board.
  - **BE0–BE3** - access each bank independently.

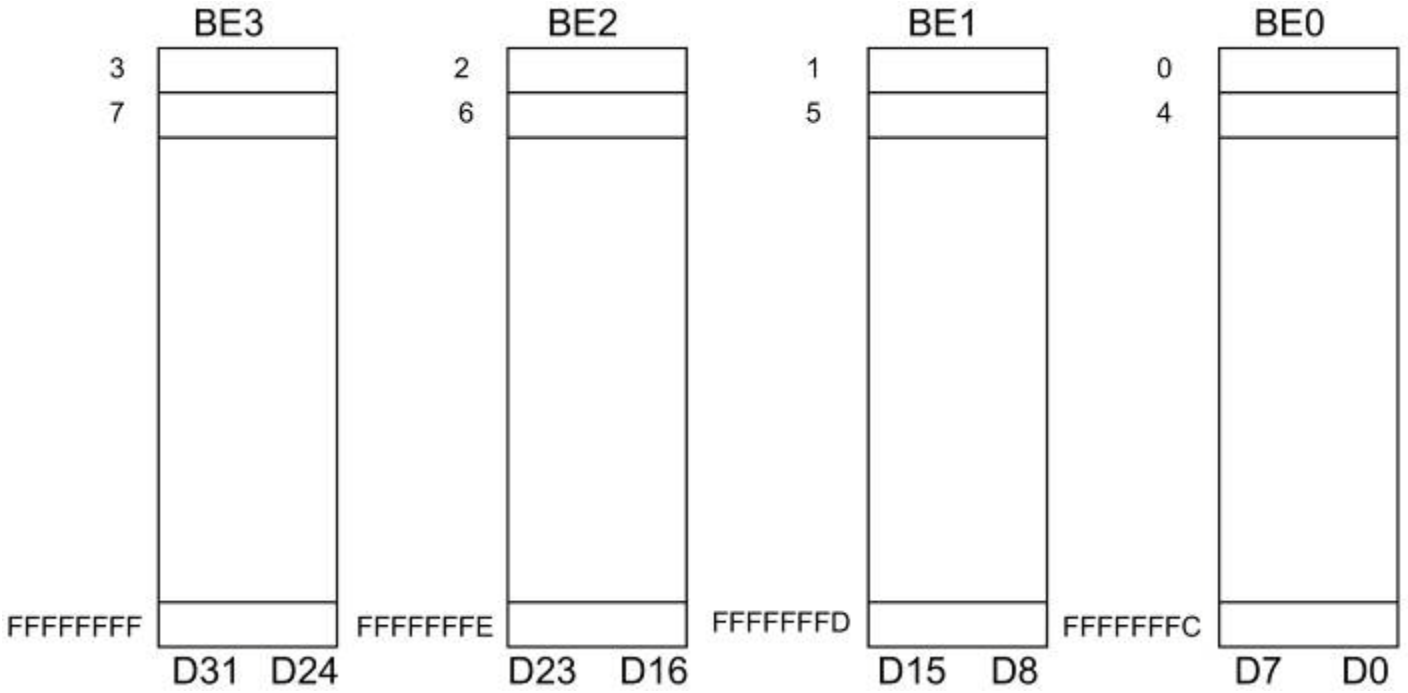


Fig. 21-4
80386 Banks

21.2: 80386: A HARDWARE VIEW
overview of pin functions

Table 21-4: Intel 386 DX PGA Pinout

Signal/Pin		Signal/Pin		Signal/Pin		Signal/Pin		Signal/Pin		Signal/Pin	
A2	C4	A24	L2	D6	L14	D28	M6	Vcc	C12	Vss	F2
A3	A3	A25	K3	D7	K12	D29	P4	Vcc	D12	Vss	F3
A4	B3	A26	M1	D8	L13	D30	P3	Vcc	G2	Vss	F14
A5	B2	A27	N1	D9	N14	D31	M5	Vcc	G3	Vss	J2
A6	C3	A28	L3	D10	M12	D/C#	A11	Vcc	G12	Vss	J3
A7	C2	A29	M2	D11	N13	ERROR#	A8	Vcc	G14	Vss	J12
A8	C1	A30	P1	D12	N12	HLDA	M14	Vcc	L12	Vss	J13
A9	D3	A31	N2	D13	P13	HOLD	D14	Vcc	M3	Vss	M4
A10	D2	ADS#	E14	D14	P12	INTR	B7	Vcc	M7	Vss	M8
A11	D1	BE0#	E12	D15	M11	LOCK#	C10	Vcc	M13	Vss	M10
A											

See the entire pinout table on page 541 of your textbook.

21.2: 80386: A HARDWARE VIEW
overview of pin functions

- **ADS, BS16, NA,** and **READY -** bus control signals.
  - ADS (address status)
  - BS16 (bus size)
  - NA (next address request)
- **CLK2** - provides the timing for the 386.
  - Always twice the system frequency.

Example 21-10

A 80386 system is advertised as 33 MHz. What frequency is connected to CLK2?

Solution:

CLK2 = 66 MHz because the frequency connected to CLK2 is always twice the system frequency.

21.2: 80386: A HARDWARE VIEW
overview of pin functions

- **W/R, D/C,** and **M/IO** - provide the bus cycle definitions and the type of the bus cycle.

Table 21-5: 80386 Bus Cycle Definition

$M/\overline{IO}$	$D/\overline{C}$	$W/\overline{R}$	Bus Cycle Type
0	0	0	Interrupt acknowledge
0	0	1	Does not occur
0	1	0	Data read (I/O)
0	1	1	Data write (I/O)
1	0	0	Memory code read
1	0	1	Halt (shutdown)
1	1	0	Memory data read
1	1	1	Memory data write

21.2: 80386: A HARDWARE VIEW
RESET

- A level-sensitive input signal into the 80386.
  - When a *low-to-high* signal is applied, 80386 will suspend all operations & registers are initialized to fixed values.

The RESET state of EIP & CS and the state of A31–A2 & BE0–BE3 have major implications about where boot ROM should be located.

Item	Contents
CS	F000
EIP	0000FFF0
A31 – A2	All high
$\overline{BE3}$ – $\overline{BE0}$	All low

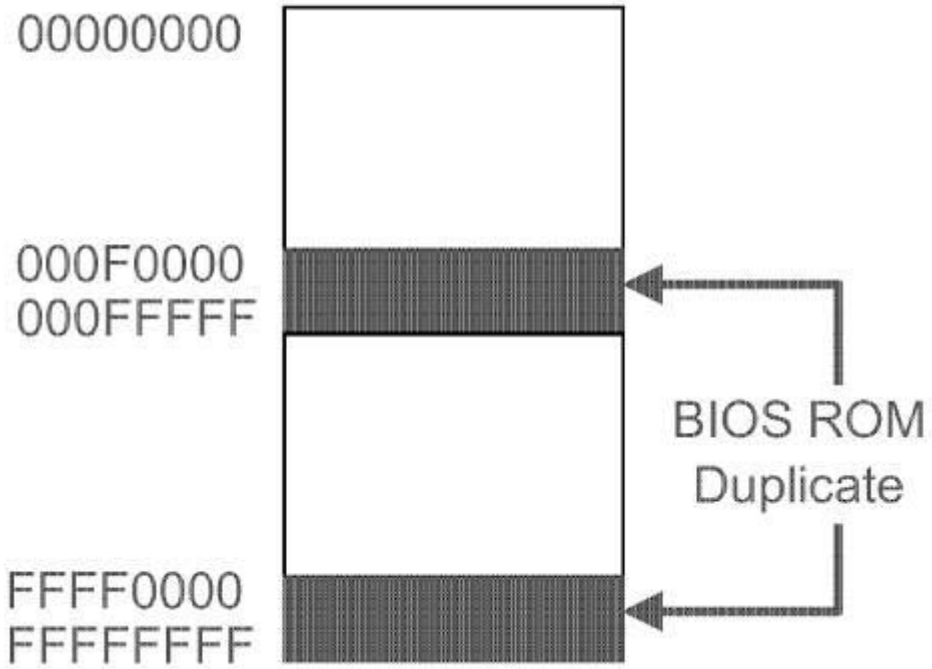
- The microprocessor will fetch the first opcode from memory location FFFFFFF0.
  - 16 bytes from the 4-gigabyte maximum address range of FFFFFFFFH.

21.2: 80386: A HARDWARE VIEW
RESET

- At this location is either a JMP FAR or a CALL FAR.
  - On executing JMP or CALL, 386, 486, & Pentium® make A31–A20 all zero, forcing it to stay within the 1-megabyte address range of real mode.

These processors wake up in *real* mode, but the first opcode address is located in *extended* memory space.

To resolve this, 386 & higher systems have duplicate ROM in both the 4-gigabyte and 1-megabyte address spaces.



21.2: 80386: A HARDWARE VIEW
bus bandwidth in the 386

- With zero wait states, it takes the 386 two clocks to perform the read or write cycle.
  - A 2-clock bus cycle is standard in high-performance processors, including RISC processors, and leads to a very high bus bandwidth.

The two clocks of the 386 memory (or I/O) bus cycle time for zero wait states are shown.

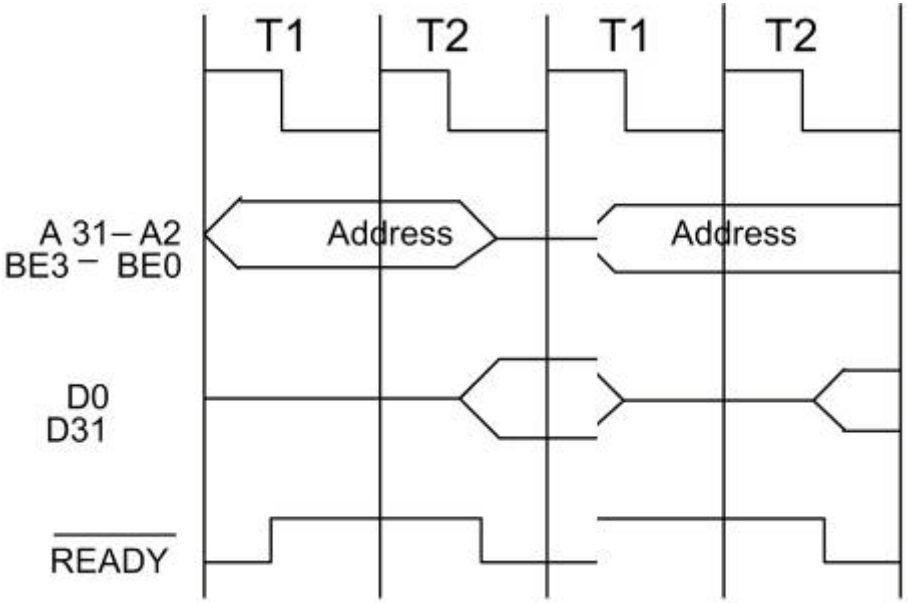


Fig. 21-7 Bus Cycle Time (Nonpiplined)

21.2: 80386: A HARDWARE VIEW
bus bandwidth in the 386

Example 21-11

Calculate the 386 bus bandwidth of a 33-MHz system with each of the following.
(a) 0 WS (b) 1 WS

Solution:

With the T state of 30 ns (1/33 MHz = 30 ns), we have memory cycle time of 60 ns and 90 ns for (a) and (b), respectively.
(a) The bus bandwidth is (1/60 ns) × 4 = 66.66 megabytes/second.
(b) The bus bandwidth is (1/90 ns) × 4 = 44.44 megabytes/second.

21.2: 80386: A HARDWARE VIEW
data misalignment in the 386

- The case of misaligned data has a major effect on the 386 bus performance.
  - For every memory read cycle the 80386 brings in 4 bytes of data using the D31–D0 data bus.
    - Referred to as doubleword alignment.

21.2: 80386: A HARDWARE VIEW
data misalignment in the 386

To make data doubleword aligned, least significant digits of the hex addresses must be 0, 4, 8, or C H.

Example 21-12

Show the data transfer of the following cases and indicate the memory cycle time if the system frequency is 25 MHz. Assume that EAX = 4598F31EH and the system is in real mode.
(a) MOV [2950],EAX (b) MOV [299A],EAX

Solution:

The system frequency of 25 MHz makes the cycle time 80 ns (1/25 MHz = 40 ns and each memory cycle is 2 clocks, giving 80 ns).

(a) In this instruction, the 4-byte content of EAX is moved to memory location with starting offset address of 2950H on the 32-bit data bus of D31–D0. This address is doubleword aligned since the least significant digit is 0. Therefore, it takes only one memory cycle or 80 ns to transfer the data.
(b) In the first memory cycle, locations with addresses of 2998H, 2999H, 299AH, and 299BH are accessed, but only 299AH and 299BH are used for storing AL and AH. In the second memory cycle, the address offsets of 299CH, 299DH, 299EH, and 299FH are accessed where only 299CH and 299D are used to store the upper 16 bits of EAX. This means that we have a total of 160 ns. If possible, this must be avoided since nonaligned data slows the data access.

21.2: 80386: A HARDWARE VIEW
I/O address space in the 386

- 80386 can access a maximum of 65,536 input ports and 65,536 output ports using IN & OUT instructions.
  - Exactly like the 8088/86/286 microprocessors.

21.3: 80386 PROTECTED MODE
protection mechanism in the 386

- In protected mode, the physical address of blocks of data or code is held by a look-up table.
  - The segment register is used as an index to a look-up table holding the physical address of the operand or code.
- 80386 provides resources to the operating system to prevent the user from accidentally or maliciously taking over the operating system core. (*kernel*)
- 80386 provides protection by allowing any data or code to be assigned a privilege level.
  - The four privilege levels are 0, 1, 2, and 3.
    - Where level 0 is the highest and level 3 is the lowest.

21.3: 80386 PROTECTED MODE
virtual memory

- A CPU with virtual memory is fooled into thinking it has access to an unlimited amount of physical (DRAM) memory. (called main memory)
- When the CPU looks for information, the OS will first search main DRAM memory.
  - If it is not there, it will bring it into RAM from secondary memory (hard disk).
  - If there is no room in RAM, the OS swaps data out of RAM and make room for new data.
    - Data swapped out depends on how the OS is designed.

21.3: 80386 PROTECTED MODE
virtual memory

- Microsoft Windows 2000, XP & Vista, all variations of Unix, Sun Microsystems' Solaris, and Mac OS X use the capability of the 80386's virtual memory.
- Two methods are used to implement virtual memory:
  - **Segmentation** - the size of the data swapped in & out can vary from 1 byte to a few megabytes.
    - In segmentation virtual memory, segment registers are selectors to a descriptor table, where information about a given piece of data & code is kept.
  - **Paging** - where the size is a multiple of one page of 4096 (4K) bytes, and prevents memory fragmentation.

21.3: 80386 PROTECTED MODE
segmentation descriptor table information

- 4 bytes for the A0–A31 address, where the code (or data) is located in main memory.
  - Allows the 386 to access any memory location in its 4-gigabyte address space.
  - A23–A0 is provided by bytes 2, 3, and 4.
  - A31–A24 is provided by byte 7.
- L0–L19: This 20-bit limit is used for checking the segment size and is limited to 1 megabyte.
  - Bytes 0 and 1 provide L0–L15.
  - D0–D3 of byte 6 is set aside for L16–L19.
- In the 386, the segment limit can be raised to 4 gigabytes.
  - With G (granularity) bit is set to *high*.
- The access byte assigns the privilege levels.

21.3: 80386 PROTECTED MODE
segmentation descriptor table information

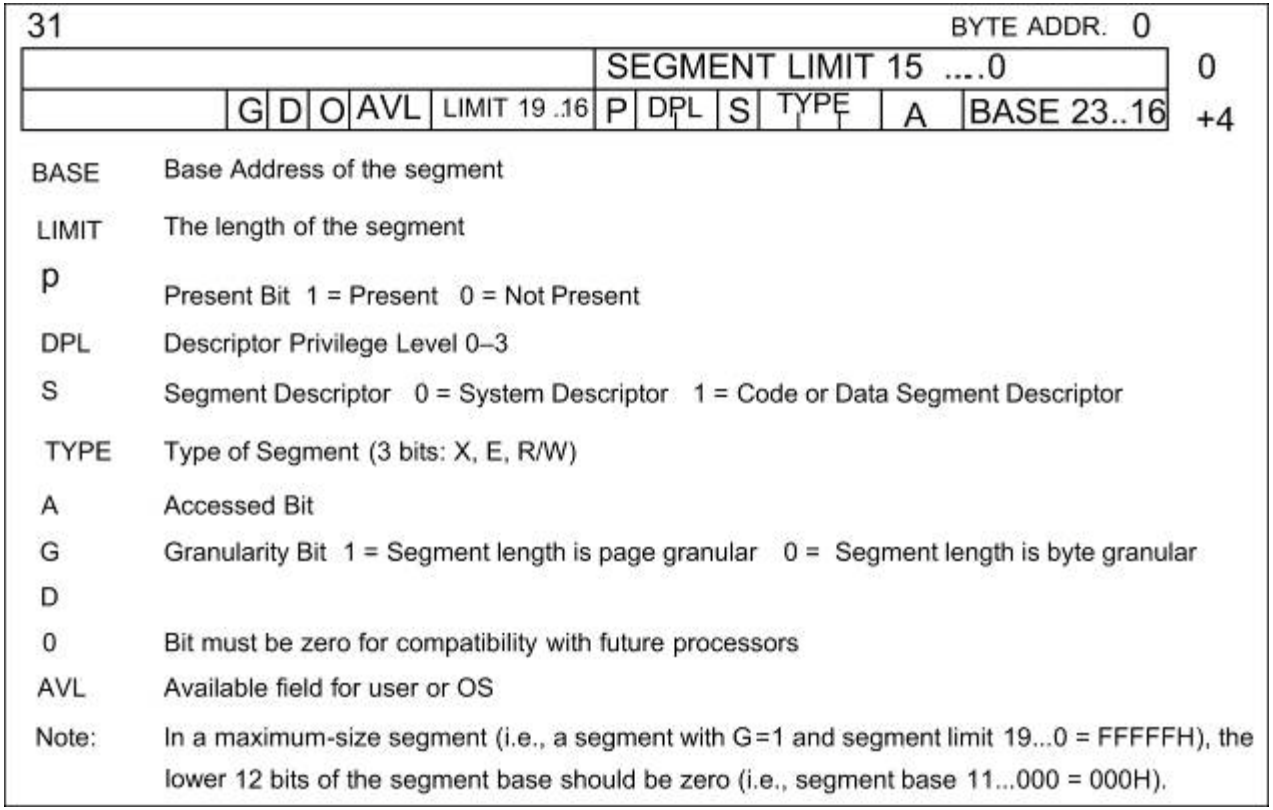


Figure 21-8
Descriptor Table Entry

21.3: 80386 PROTECTED MODE
D0-D7 of the access byte

- **A** (accessed) bit
  - If data or code is accessed (used), A = 1; else A = 0.
    - Allows the OS to monitor the A bit periodically to see if the CPU is using this piece of code or data.
- **R/W** (read/write) bit - allows code or data to be read protected or write protected.
  - The OS core can be write protected, which prevents the user from writing into it and crashing the system.
- **X** bit - different meanings for data & code segments.
  - For data, it indicates if the segment should expand down as the stack segment grows, or up as the data grows.
  - For code, it is used to enforce privilege level access.

21.3: 80386 PROTECTED MODE
D0-D7 of the access byte

- **E** bit - indicates if the information is executable or nonexecutable
  - E = 1 (executable code); E = 0 (data and stack)
    - Also affects the way the **X** and **R/W** bits are interpreted.
- **S** bit - indicates if the descriptor belongs to the code & data segment (S = 1) or system segment (S = 0).
- **DPL** (descriptor privilege level) bits - allows one of the combinations, 00, 01, 10, or 11, to be assigned to the code or data, indicating the privilege level.

21.3: 80386 PROTECTED MODE
D0-D7 of the access byte

- **P** (present) bit - indicates if the piece of code or data is present in main memory (DRAM).
  - If present (P = 1), the CPU will process it.
  - If present (P = 0), the CPU causes an exception & bring the code or data into main memory from the hard disk.
    - Then sets P = 1 to indicate the information is now present.

21.3: 80386 PROTECTED MODE
local and global descriptor tables

- There are two types of descriptor tables for the 386:
  - Local descriptor table (**LDT**) for individual tasks.
  - Global descriptor table (**GDT**) used for the system.

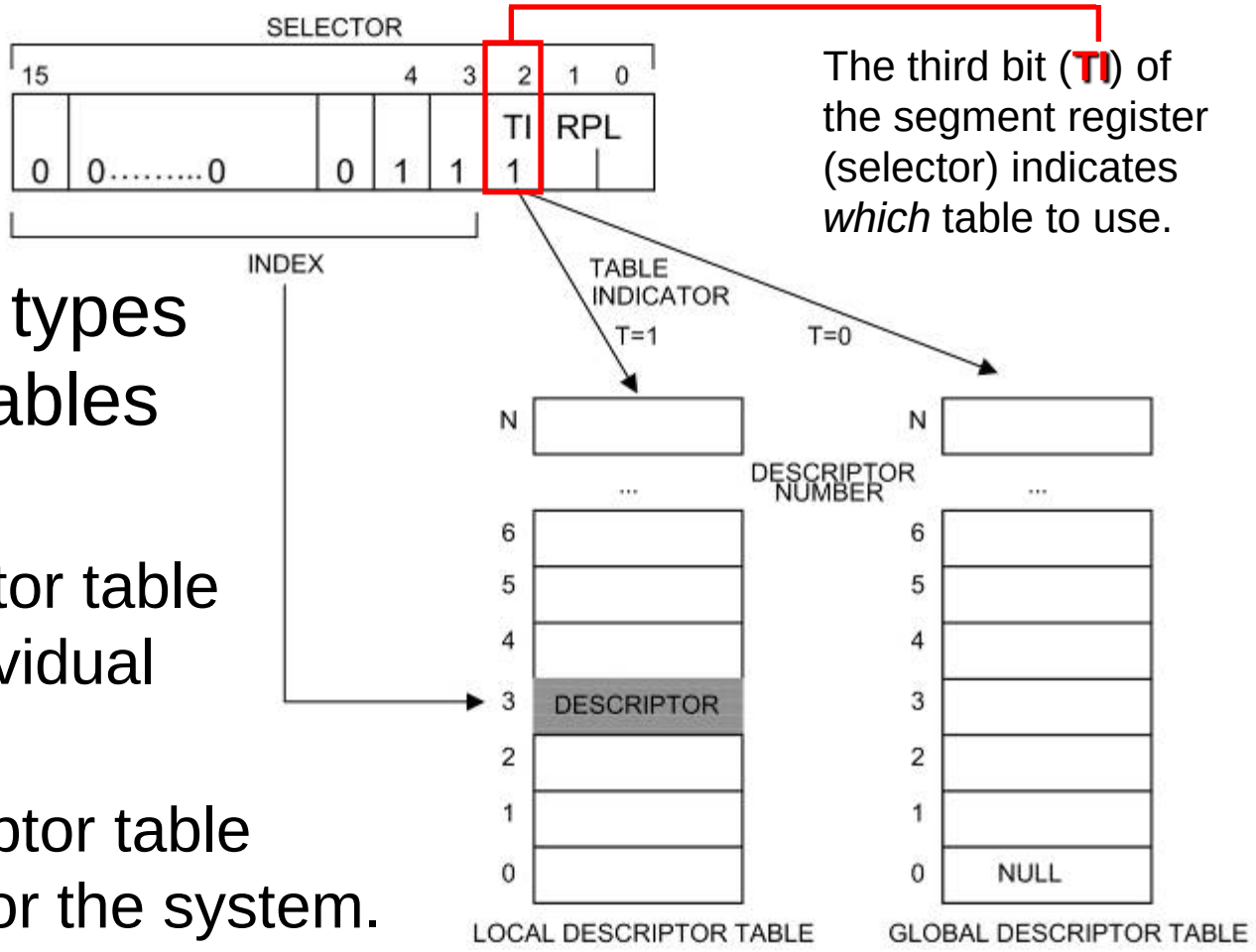


Figure 21-9 LDT and GDT Selection

21.3: 80386 PROTECTED MODE
64 terabytes of virtual memory

- The 386 can access 64 terabytes of hard disk (virtual memory) as long as the virtual memory is broken down into 4-gigabyte pieces.
  - Since it has only 32 address pins.
- A drawback of 386 segmentation is variable segment size, leading to memory fragmentation.
  - Another is the absence of what is called a *dirty bit* in the access byte of the descriptor table.
- Problems of variable segment size and lack of a dirty bit in segmentation are fixed in the paging method of virtual memory.

21.3: 80386 PROTECTED MODE
paging

- In paging virtual memory, main memory is divided into fixed 4K-byte chunks instead of variable sizes.
  - 80386 keeps a table, the *translation lookaside buffer* (TLB) of the 32 most recent pages present in main memory to prevent swapping unnecessarily.
- The 32-bit address of the operand in the 386 is called the linear address.
  - It can be a direct value, or pointed to by registers.
- This address must be translated into a physical address to be put on the A31–A0 address pins.
  - Sent out for the address decoder to find the location.

21.3: 80386 PROTECTED MODE
paging

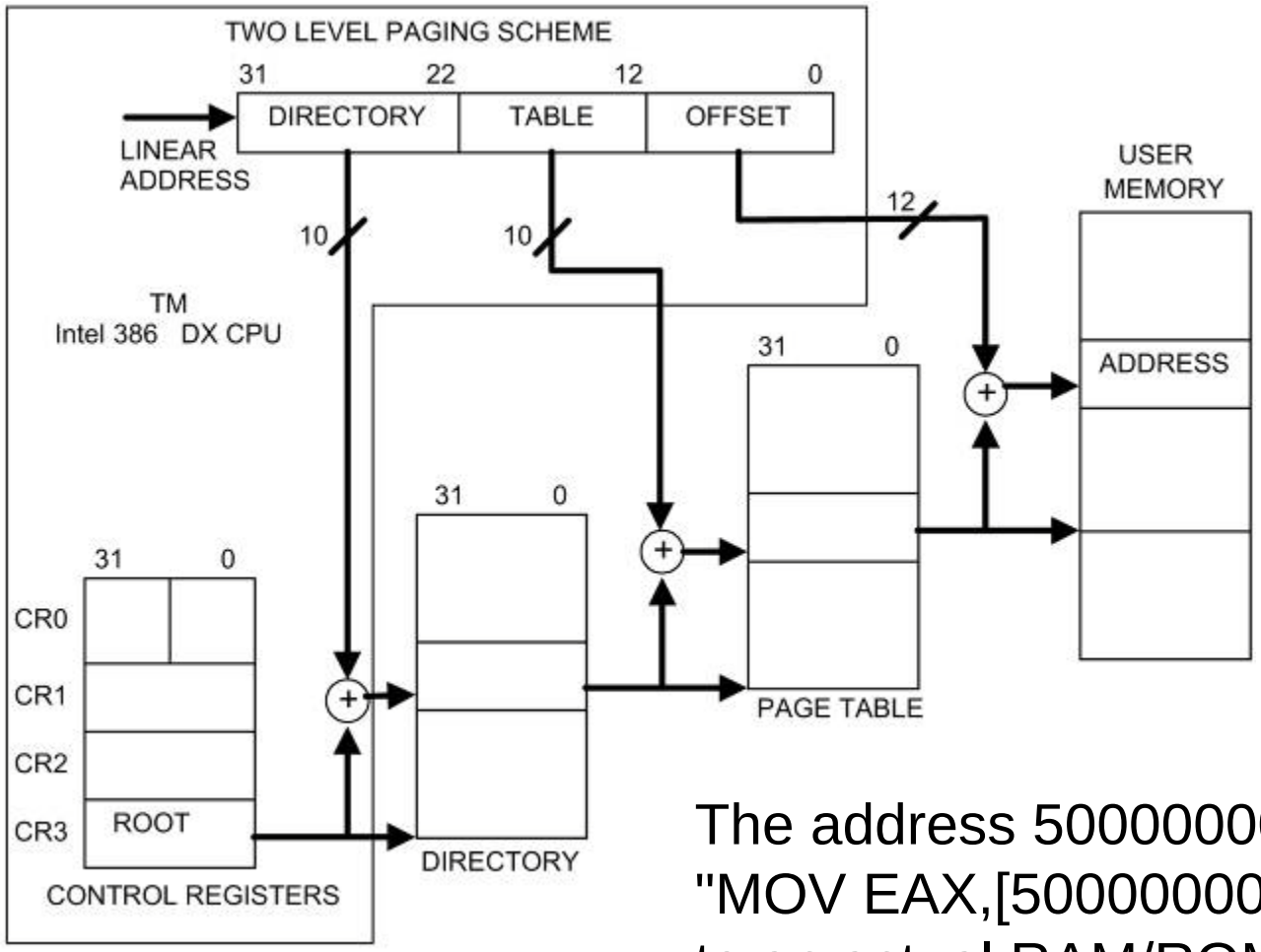


Fig. 21-10 Paging Mechanism

The address 50000000H in instruction "MOV EAX,[50000000]" does not refer to an actual RAM/ROM address

21.3: 80386 PROTECTED MODE
a linear address to a physical address

- In paging, the linear address is divided into 3 parts.
  - The upper 10 bits (A31–A22) are used for an entry into what is called a page directory.
    - There is a 32-bit register, **CR3**, inside the 386 that holds the physical base address of the page directory.
  - A21–A11 (10 bits total) of the linear address are used to point to one of the page table entries.
    - Each entry in this second table has 4 bytes.
  - The lower 12 bits of the physical address are the lower 12 bits of the linear address.

21.3: 80386 PROTECTED MODE
a linear address to a physical address

- Only the lower 12 bits of the linear address match the lower 12 bits of the physical location in RAM or ROM where data is located.

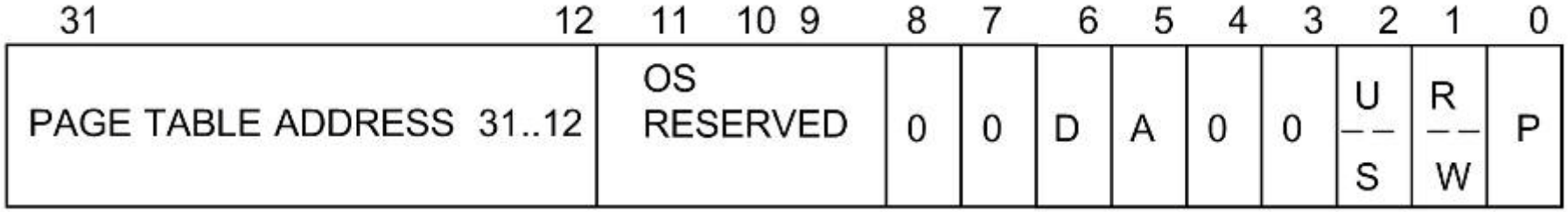


Fig. 21-11 Page Directory Entry (Points to Page Table)

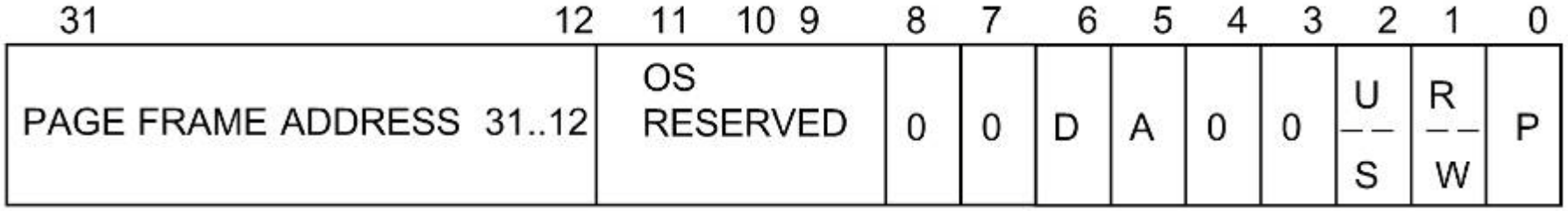


Fig. 21-12 Page Directory Entry (Points to Page)

21.3: 80386 PROTECTED MODE
the bigger the TLB, the better

- The 386 TLB keeps the list of 32 most recently used pages, allowing CPU access to 128K of code & data.

Without converting linear addresses to physical addresses (two-stage table translation)

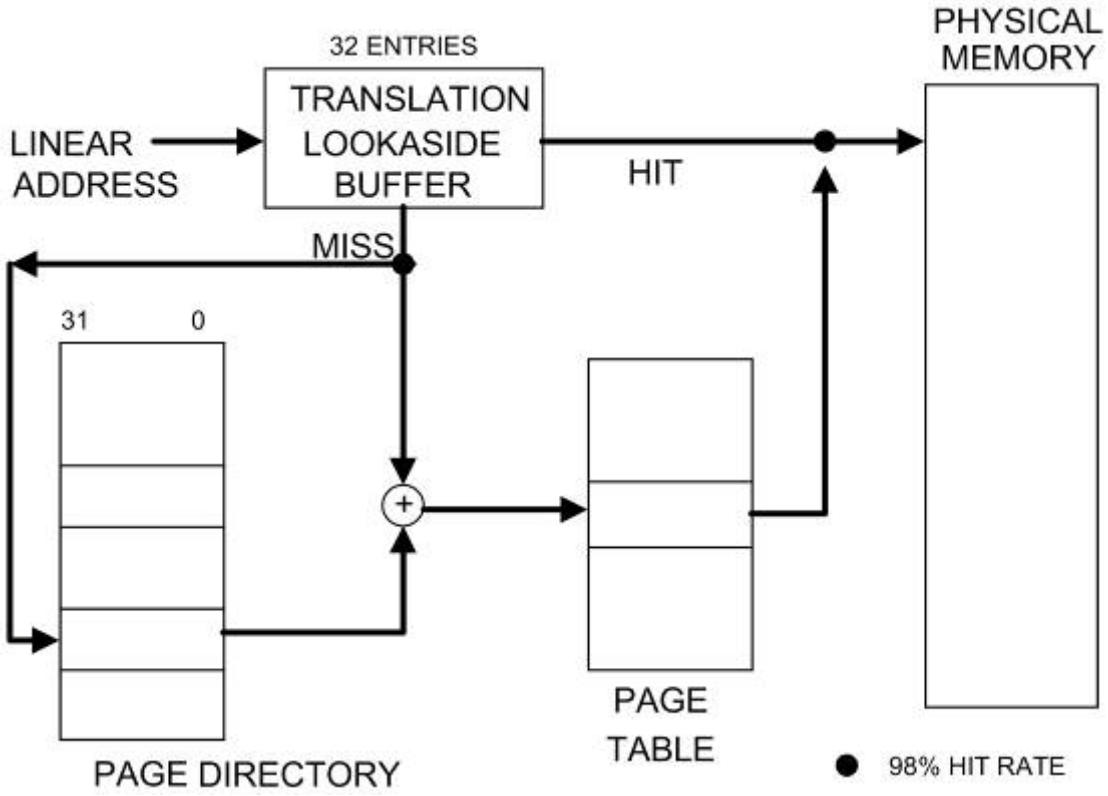


Fig. 21-13 Translation Lookaside Buffer)1

21.3: 80386 PROTECTED MODE
the bigger the TLB, the better

One way to enhance the processor is to increase the number of pages held by the TLB.

Table 21-7: Paging and Segmentation Comparison

Feature	Paging	Segmentation
Size	4K bytes	Any size
Levels of privilege	2	4
Base address	4K-byte aligned	Any address
Dirty bit	Yes	No
Access bit	Yes	Yes
Present bit	Yes	Yes
Read/write protection	Yes	Yes

21.3: 80386 PROTECTED MODE
virtual 8086 mode

- A major dilemma for designers was how to enhance the 386 & run 8088/86 software in protected mode.
  - Solved by adding the virtual 8086 mode to the 386.
    - The 386 partitions memory into 1-megabyte sections, each assigned to one task.
  - In virtual 8086 mode, 386 uses the SEG:OFFSET concept used in the 8088/86 microprocessor.

Dec Hex Bin
21 15 00010101

ENDS ; TWENTY-ONE

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

The x86 PC
assembly language, design, and interfacing
fifth edition

Prentice Hall