Teaching/Examination Scheme

Savitribai Phule Pune University Second Year of Engineering (2019 Course) 210254: Microprocessor		
Teaching Scheme	**Credit Scheme**	**Examination Scheme and Marks**
Lecture: 03 Hours/Week	**03**	**Mid_Semester(TH): 30 Marks** **End_Semester(TH): 70 Marks**
Prerequisite Courses :	210248: Digital Electronics and Logic Design	
Companion Course :	210258: Microprocessor Laboratory	

Unit 1:
Introduction to 80386

Unit I	Introduction to 80386	(07 Hours)
Brief History of Intel Processors, 80386 DX Features and Architecture, Programmers Model, Operating modes, Addressing modes and data types. **Applications Instruction Set**: Data Movement Instructions, Binary Arithmetic Instructions, Decimal Arithmetic Instructions, Logical Instructions, Control Transfer Instructions, String and Character Transfer Instructions, Instructions for Block Structured Language, Flag Control Instructions, Coprocessor Interface Instructions, Segment Register Instructions, Miscellaneous Instructions.		
#Exemplar/Case Studies	Study-Evolution of Microprocessor	
*Mapping of Course Outcomes for Unit I	CO1,CO2	

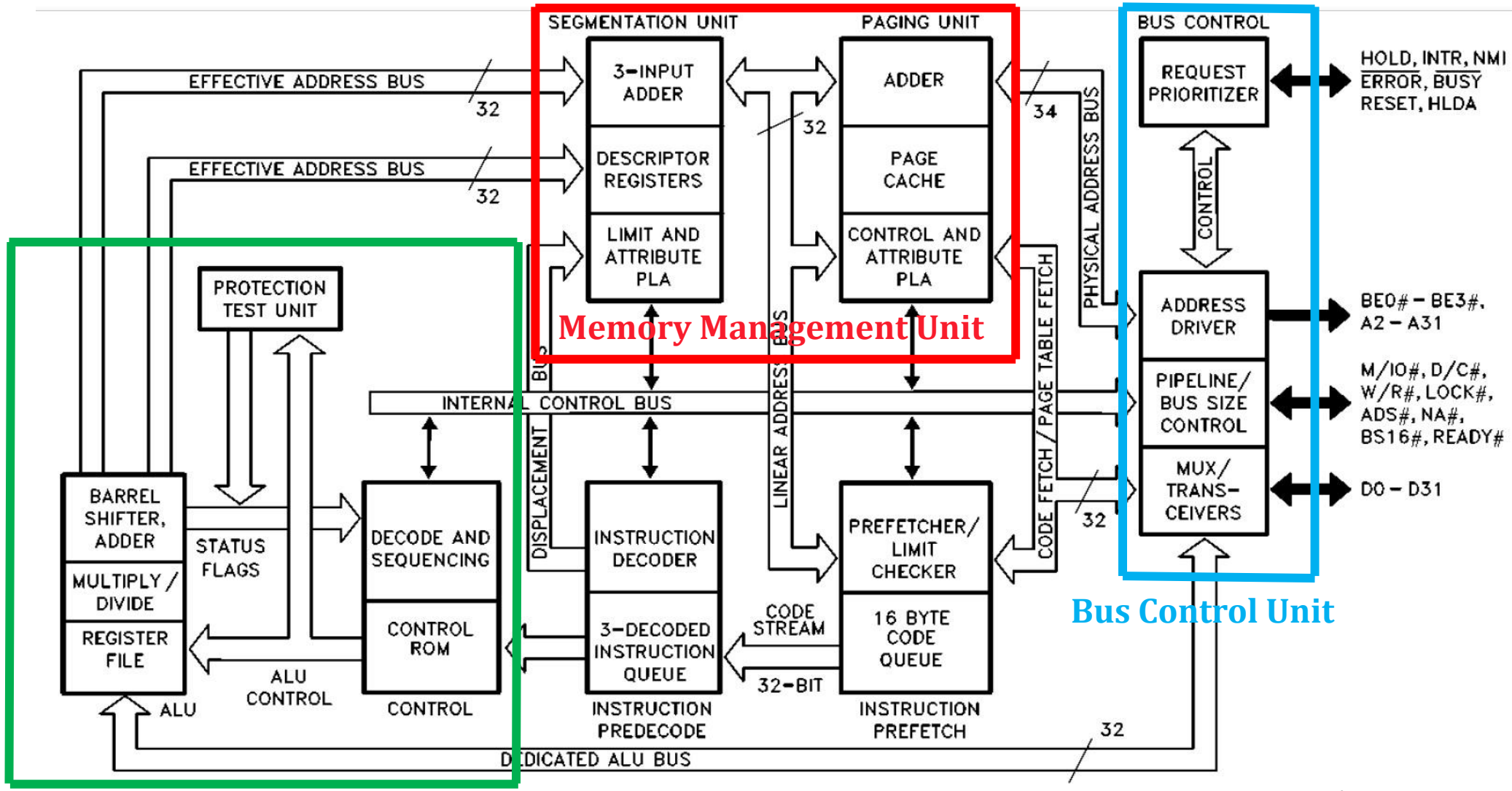
Unit 1:
Introduction to 80386

80386 DX Features and Architecture, Programmers Model

Features of 80386DX

- Flexible **32-bit Microprocessor**
  - 8, 16, 32-Bit Data Types
  - 8 General Purpose 32-Bit Registers
  - **32 bit ALU**
  - **32 bit Data Bus**(4 memory Bank)
  - **32 bit Address Bus**(4 GB Memory)
- Very Large Address Space
  - 4 Gigabyte Physical
  - **64 Terabyte Virtual**
  - 4 Gigabyte Maximum Segment Size
- Integrated Memory Management Unit
  - Segmentation and Paging
  - **4 Levels of Protection**
  - Fully Compatible with 80286
- Hardware Debugging Support
- **3 stage Pipeline**
- **Multitasking**
- **Operating Speed-16,20,25,33 MHz**
- SX (16 bit Data Bus)
  DX(32 bit Data Bus)

80386DX Architecture



80386DX Architecture

The Intel386DX consists of

- A central processing unit,
  - Execution unit
  - Instruction unit
- A memory management unit and
  - Segmentation unit
  - Paging unit
- A bus interface.
  - offers address pipelining, dynamic data bus sizing, and direct Byte Enable signals for each byte of the data bus

Execution Unit

- The execution unit contains the
  - ✔ Eight 32-bit general purpose registers which are used for both address calculation, data operations and
  - ✔ A 64-bit barrel shifter used to speed shift, rotate, multiply, and divide operations.
- The multiply and divide logic uses a 1-bit per cycle algorithm.
- The multiply algorithm stops the iteration when the most significant bits of the multiplier are all zero.
- This allows typical 32-bit multiplies to be executed in under one microsecond.

Execution Unit (Cont…)

✔ The linear address consists of two components:
- The segment base address and
- An effective address.

✔ The effective address is calculated by using four address elements:
- **DISPLACEMENT:** An 8-, 16- or 32-bit immediate value
- **BASE:** The contents of any general purpose register. It is generally used by compilers to point to the start of the local variable area.
- **INDEX:** The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters.
- **SCALE:** The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. Scaled index mode is especially useful for accessing arrays or structures.

EA = Base Register + (Index Register * Scaling) + Displacement.

Instruction and Segmentation Unit

Instruction Unit:

The instruction unit decodes the instruction opcodes and stores them in the decoded instruction queue for immediate use by the execution unit.

Segmentation Unit:

Segmentation allows the managing of the logical address space by providing an extra addressing component, one that allows

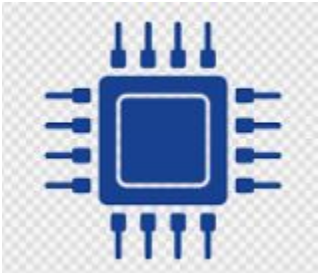
- ✔ easy code
- ✔ data relocatability, and
- ✔ efficient sharing

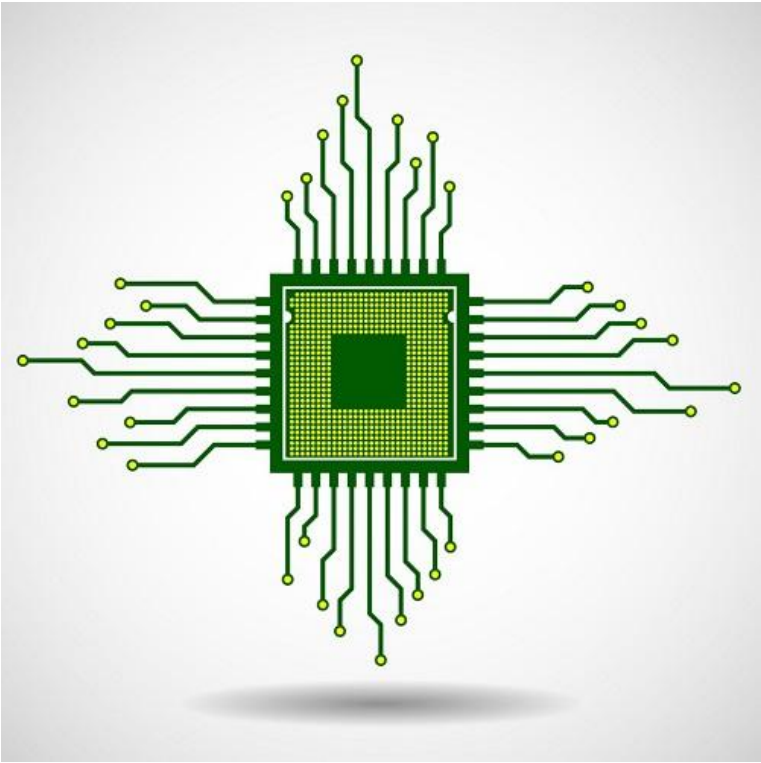
Paging Unit

- The paging mechanism operates beneath and is transparent to the segmentation process, to allow management of the physical address space.

- Each segment is divided into one or more 4K byte pages.

Microprocessor





- ✔ **Unit 2: Bus Cycles and System Architecture**

Unit 2:
Bus Cycles and System Architecture

Unit II	Bus Cycles and System Architecture	(07 Hours)
Initialization- Processor State after Reset. Functional pin Diagram, functionality of various pins, I/O Organization, Memory Organization (Memory banks), Basic memory read and writes cycles with timing diagram. **Systems Architecture**- Systems Registers (Systems flags, Memory Management registers, Control registers, Debug registers, Test registers), System Instructions.		
#Exemplar/Case Studies	Study-Motherboard of Computer and it's components.	
*Mapping of Course Outcomes for Unit II	CO3	

Unit 2:
Bus Cycles and System Architecture

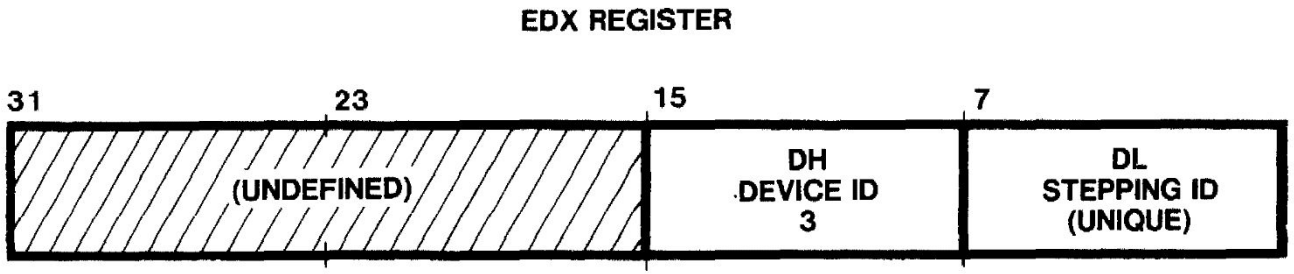
Initialization

Initialization

- After a signal on the RESET pin, certain registers of 80386 are set to predefined values.
- These values are adequate to enable execution of a bootstrap program.

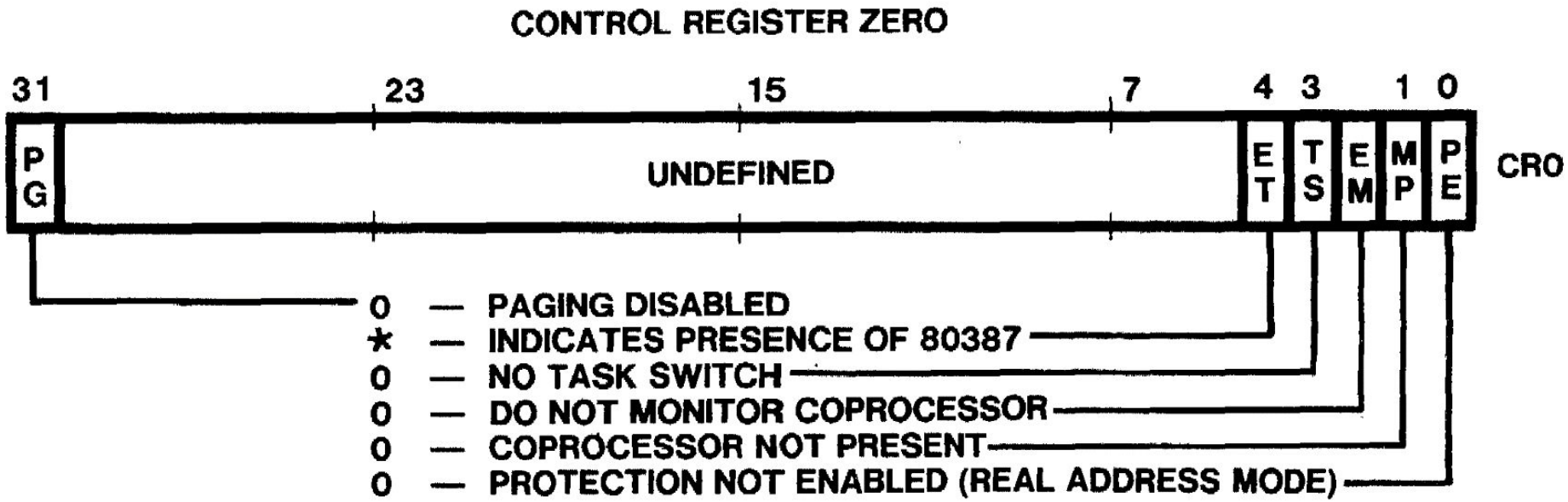
Processor State after RESET

- Contents of EAX depends on results power-up self test
- Self-test may be requested externally by assertion of BUSY# at the end of RESET (EAX=0 if the 80386 passed the test, else 80386 unit is faulty)
- If self-test is not requested , EAX is undefined
- DX holds a component identifier and revision number
  (DH=3, indicates 80386, DL=unique identifier of the revision level)



Processor State after RESET

- Control Register 0 (CR0) contains



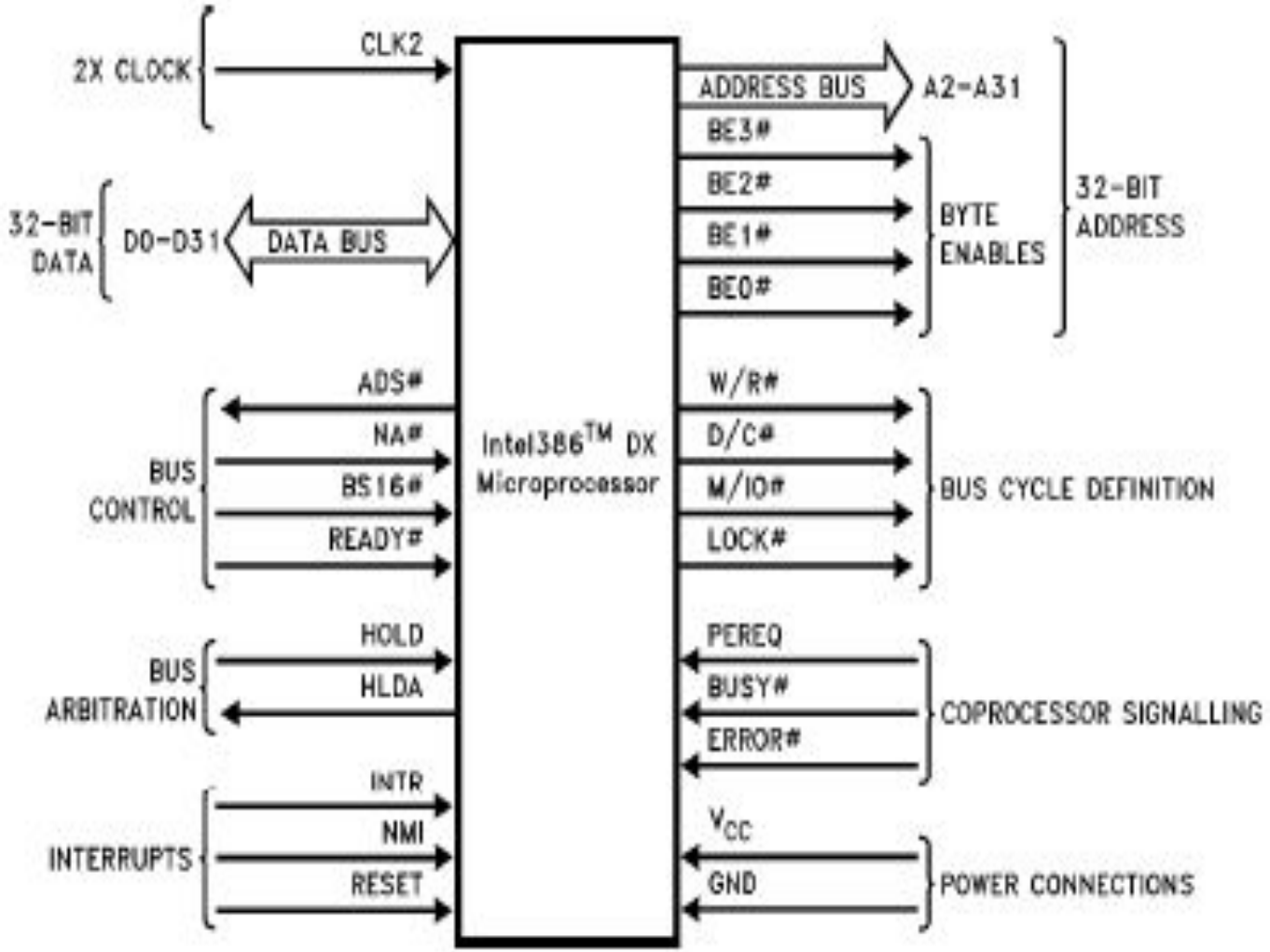
State of flags and other registers

The remaining registers and flags are set as follows:

EFLAGS	= 00000002H
IP	= 0000FFF0H
CS selector	= 000H
DS selector	= 0000H
ES selector	= 0000H
SS selector	= 0000H
FS selector	= 0000H
GS selector	= 0000H
IDTR:	
base	= 0
limit	= 03FFH

All registers not mentioned above are undefined.

Functional Pin Diagram



Pin Description Table

Symbol	Type	Name and Function
CLK2	I	**CLK2** provides the fundamental timing for the Intel386 DX.
$D_{31}-D_0$	I/O	**DATA BUS** inputs data during memory, I/O and interrupt acknowledge read cycles and outputs data during memory and I/O write cycles.
$A_{31}-A_2$	O	**ADDRESS BUS** outputs physical memory or port I/O addresses.
BE0#–BE3#	O	**BYTE ENABLES** indicate which data bytes of the data bus take part in a bus cycle.
W/R#	O	**WRITE/READ** is a bus cycle definition pin that distinguishes write cycles from read cycles.
D/C#	O	**DATA/CONTROL** is a bus cycle definition pin that distinguishes data cycles, either memory or I/O, from control cycles which are: interrupt acknowledge, halt, and instruction fetching.
M/IO#	O	**MEMORY I/O** is a bus cycle definition pin that distinguishes memory cycles from input/output cycles.
LOCK#	O	**BUS LOCK** is a bus cycle definition pin that indicates that other system bus masters are denied access to the system bus while it is active.
ADS#	O	**ADDRESS STATUS** indicates that a valid bus cycle definition and address (W/R#, D/C#, M/IO#, BE0#, BE1#, BE2#, BE3# and $A_{31}-A_2$) are being driven at the Intel386 DX pins.
NA#	I	**NEXT ADDRESS** is used to request address pipelining.
READY#	I	**BUS READY** terminates the bus cycle.
BS16#	I	**BUS SIZE 16** input allows direct connection of 32-bit and 16-bit data buses.
HOLD	I	**BUS HOLD REQUEST** input allows another bus master to request control of the local bus.

Pin Description Table (Cont…)

Symbol	Type	Name and Function
HLDA	O	**BUS HOLD ACKNOWLEDGE** output indicates that the Intel386 DX has surrendered control of its local bus to another bus master.
BUSY#	I	**BUSY** signals a busy condition from a processor extension.
ERROR#	I	**ERROR** signals an error condition from a processor extension.
PEREQ	I	**PROCESSOR EXTENSION REQUEST** indicates that the processor extension has data to be transferred by the Intel386 DX.
INTR	I	**INTERRUPT REQUEST** is a maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
NMI	I	**NON-MASKABLE INTERRUPT REQUEST** is a non-maskable input that signals the Intel386 DX to suspend execution of the current program and execute an interrupt acknowledge function.
RESET	I	**RESET** suspends any operation in progress and places the Intel386 DX in a known reset state. See **Interrupt Signals** for additional information.
N/C	—	**NO CONNECT** should always remain unconnected. Connection of a N/C pin may cause the processor to malfunction or be incompatible with future steppings of the Intel386 DX.
V_{CC}	I	**SYSTEM POWER** provides the +5V nominal D.C. supply input.
V_{SS}	I	**SYSTEM GROUND** provides 0V connection from which all inputs and outputs are measured.

I/O Addressing

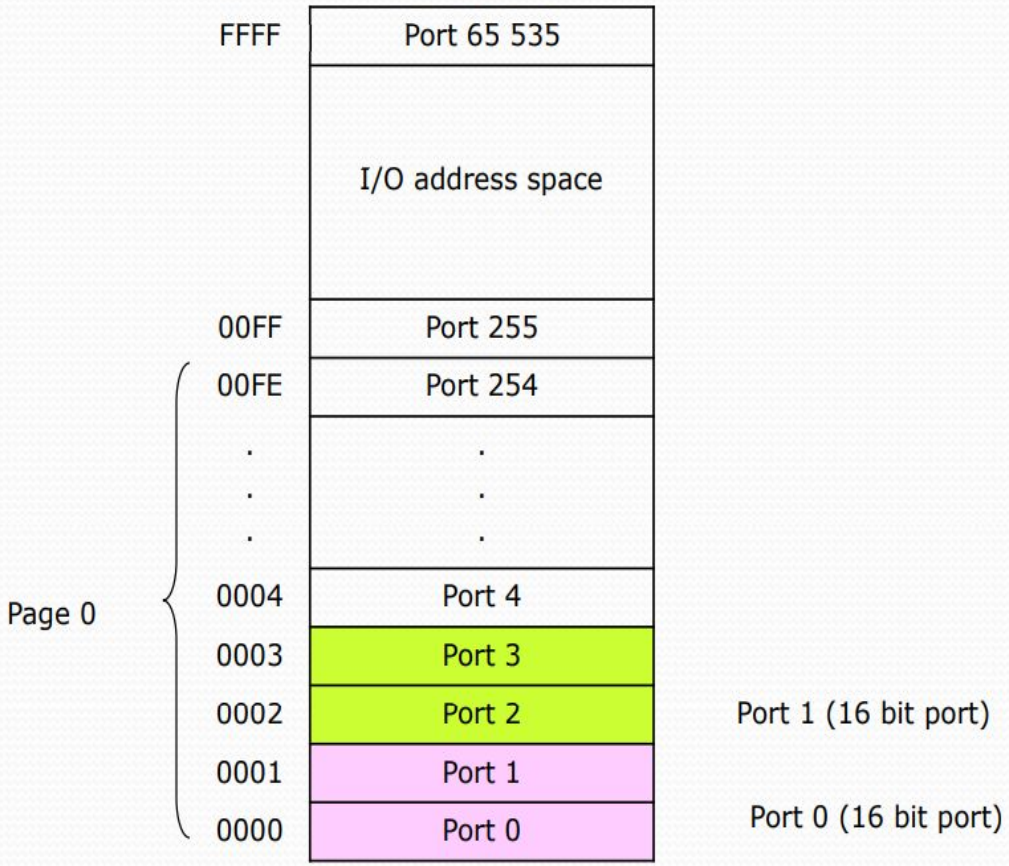
The 80386 allows input/output to be performed in either of two ways:

a. By means of a separate I/O address space (using specific I/O instructions)

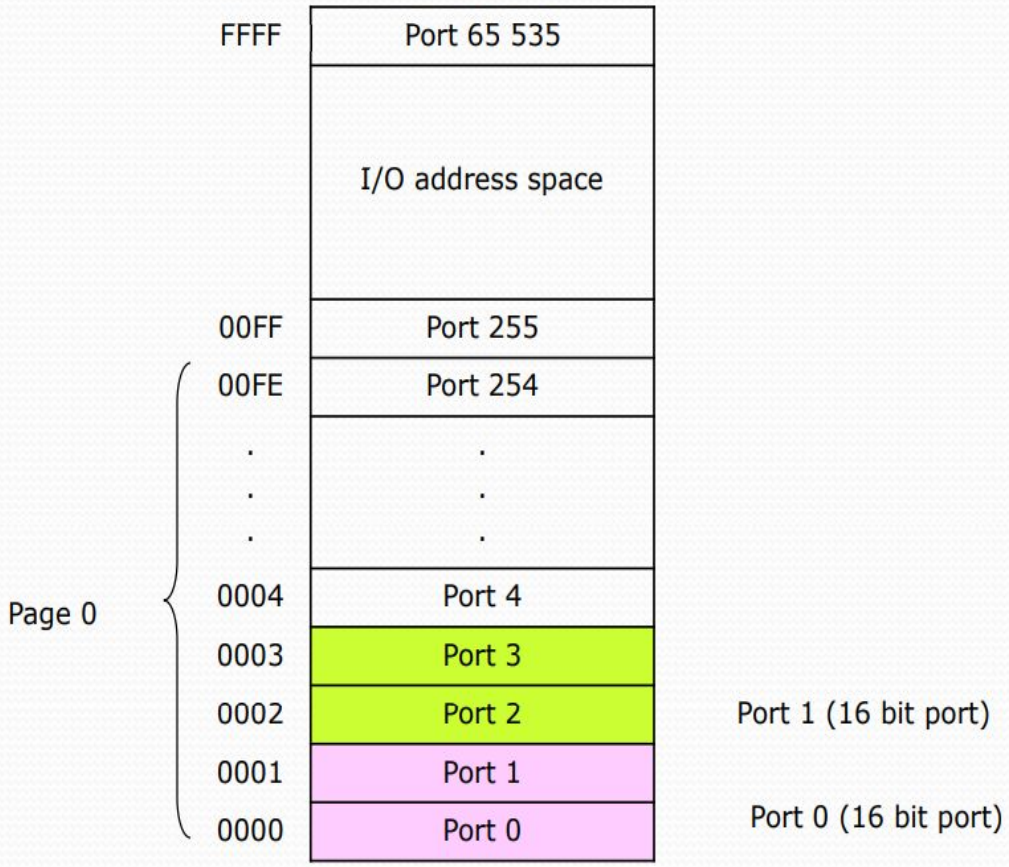
b. By means of memory-mapped I/O (using generalpurpose operand manipulation instructions)

Separate I/O address space (An Isolated I/O)

- ✔ I/O devices treated separately from memory.
  - Hardware and software architecture of 8088/8086 support separate memory I/O address space.
- ✔ Can be accessed as either byte-wide or word-wide.
- ✔ Can be treated as either independent byte-wide I/O ports or word-wide I/O ports.
- ✔ Page 0:
  - Certain I/O instructions can only perform operations to ports in this part of the address range.
  - Other I/O instructions can input/output data for ports anywhere in the address space.



Separate I/O address space (An Isolated I/O)



Advantages: -

- ✔ 1 MByte memory address space is available for use with memory.
- ✔ Special instructions have been provided in the instruction set of 8088/8086 to perform isolated I/O input and output operations.
- ✔ These instructions have been tailored to maximize I/O performance.

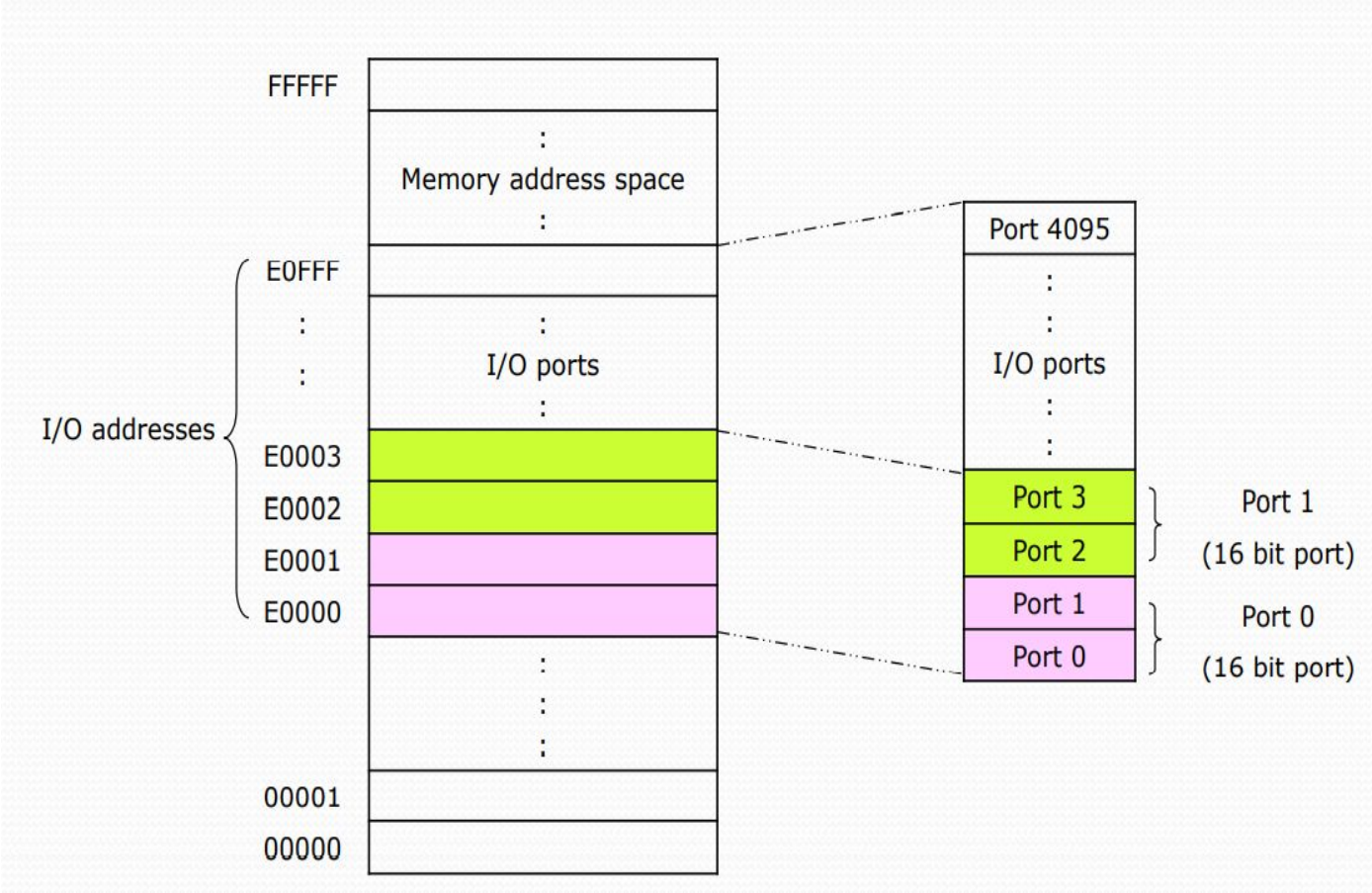
Disadvantages: -

- ✔ All input and output data transfers must take place between AL or AX register and the I/O port

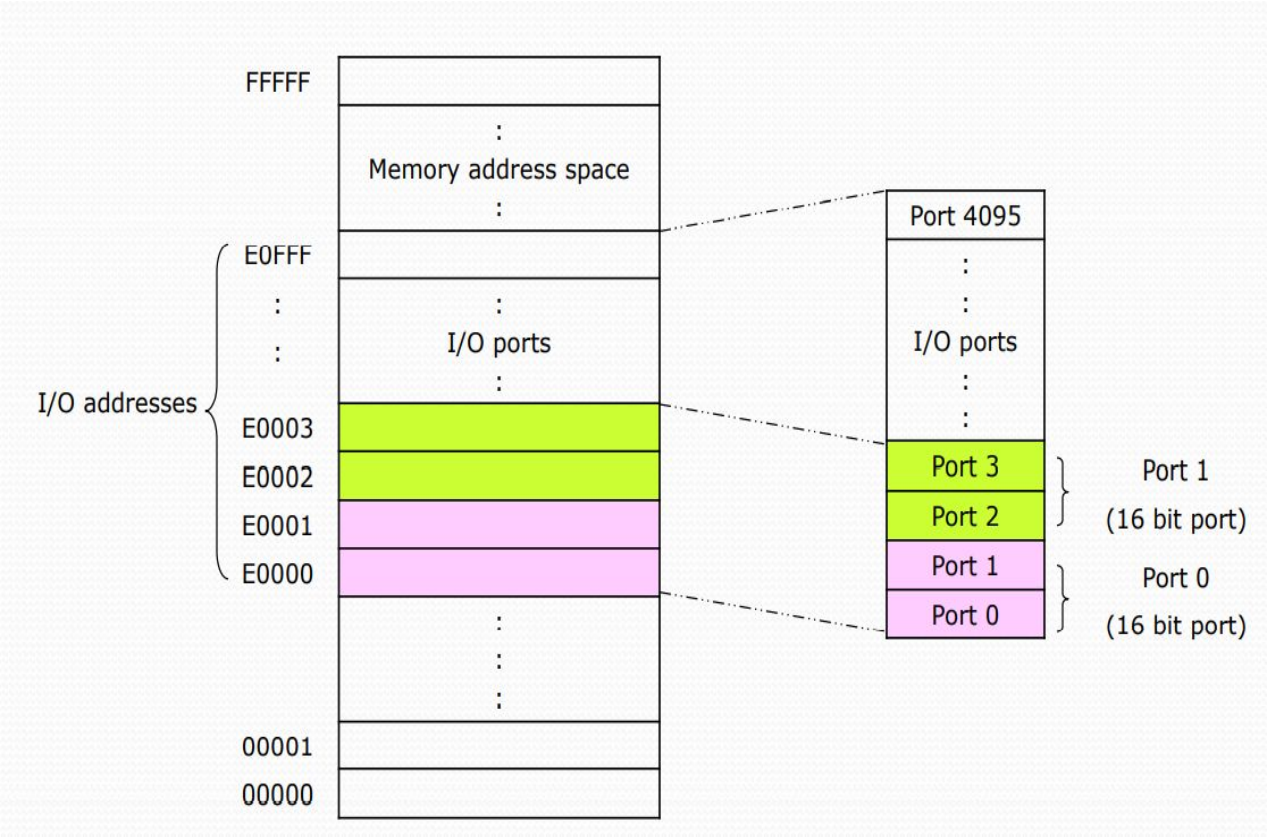
Memory-mapped I/O

I/O devices is placed in memory address space of the microcomputer.

- The memory address space is assigned to I/O devices.
- MPU looks at the I/O port as though it is a storage location in memory.
- Make use of instructions that affect data in memory rather than special input/output instructions.



Memory-mapped I/O



Advantages:

- ✔ Many more instructions and addressing modes are available to perform I/O operations.
- ✔ I/O transfers can now take place between I/O port and internal registers other than just AL/AX.

Disadvantages:

- ✔ Memory instructions tend to execute slower than those specifically designed for isolated I/O.
- ✔ Part of the memory address space is lost

I/O Instructions

There are two classes of I/O instruction:

1. Those that transfer a single item (byte, word, or doubleword) located in a register.

2. Those that transfer strings of items (strings of bytes, words, or doublewords) located in memory.

These are known as "string I/O instructions" or "block I/O instructions".

Register I/O Instructions

- The I/O instructions IN and OUT are provided to move data between I/O ports and the EAX (32-bit I/O), the AX (I6-bit I/O), or AL (8-bit I/O) general registers.
- IN and OUT instructions addresses I/O ports either directly, with the address of one of up to 256 port

Mnemonic	Meaning	Format	Operation
IN	Input direct	IN Acc, Prt	(Acc) ← (Port) Acc = AL or AX
	Input indirect (variable)	IN Acc, DX	(Acc) ← ((DX))
OUT	Output direct	OUT Prt, Acc	(Port) ← (Acc)
	Output indirect (variable)	OUT DX, Acc	((DX)) ← (Acc)

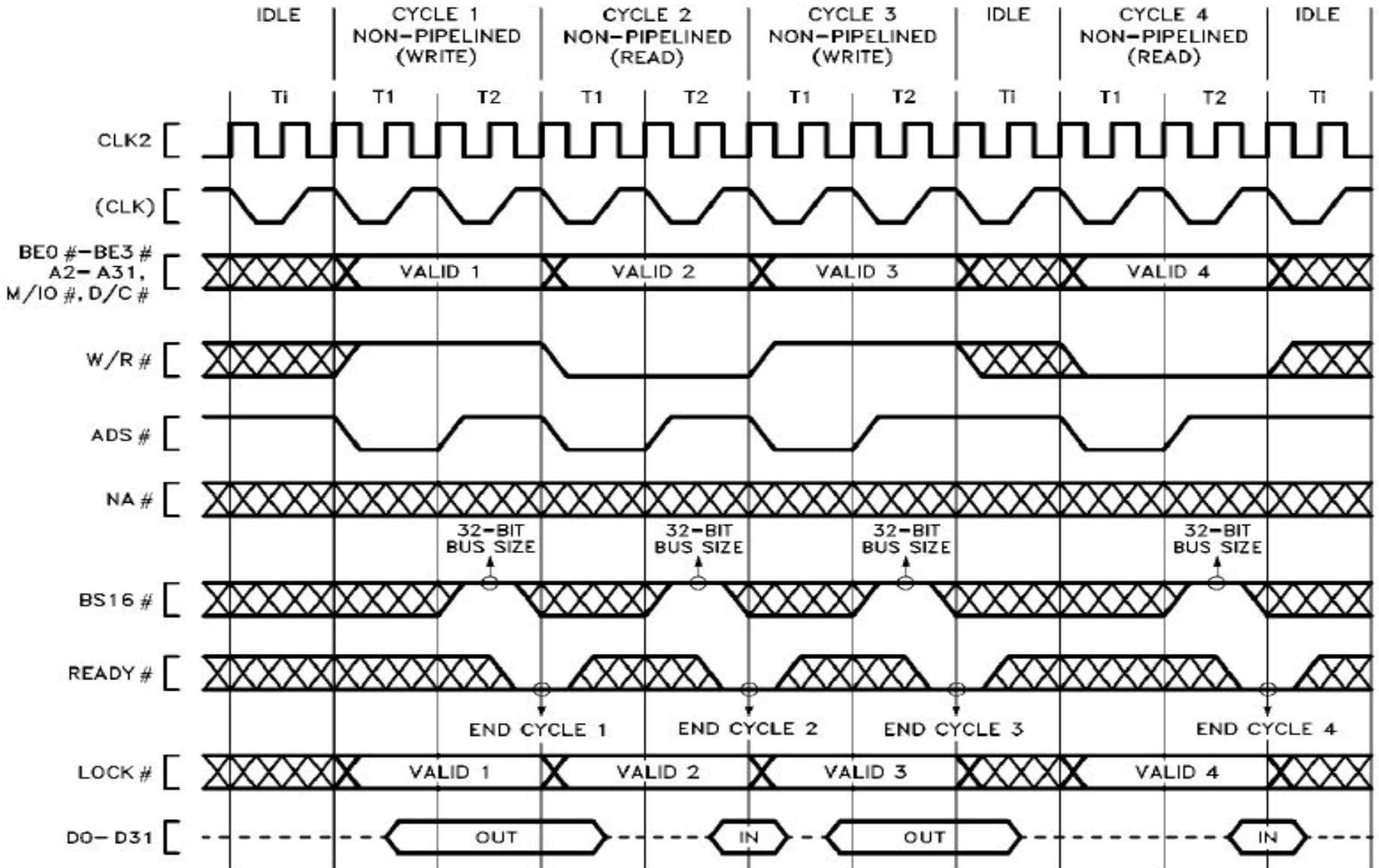
Block I/O Instructions

- The block (or string) I/O instructions INS and OUTS move blocks of data between I/O ports and memory space.
- Block I/O instructions use the DX register to specify the address of a port in the I/O address space.
- Block I/O instructions use either SI or DI to designate the source or destination memory address.
- For each transfer, SI or DI are automatically either incremented or decremented as specified by the direction bit in the flags register.

Read and Write Cycles

- Data transfers occur as a result of bus cycles, classified as read or write cycles.
- Two choices of address timing are dynamically selectable: non-pipelined, or pipelined.
- After a bus idle state, the processor always uses non-pipelined address timing.
- However, the NA# (Next Address) input may be asserted to select pipelined address timing for the next bus cycle.
- When pipelining is selected and the Intel386 DX has a bus request pending internally, the address and definition of the next cycle is made available even before the current bus cycle is acknowledged by READY#.
- Terminating a read cycle or write cycle, like any bus cycle, requires acknowledging the cycle by asserting the READY# input.
- Until acknowledged, the processor inserts wait states

Non-pipelined read & write cycles



Non-pipelined read & write cycles

- At the end of the second bus state within the bus cycle, READY# is sampled
- If asserted the bus cycle terminates
- Else the cycle continues another bus state (a wait state) and READY# is sampled again at the end of that state.
- This continues indefinitely until the cycle is acknowledged by READY# asserted.

Unit 2:
Bus Cycles and System Architecture

System Architecture

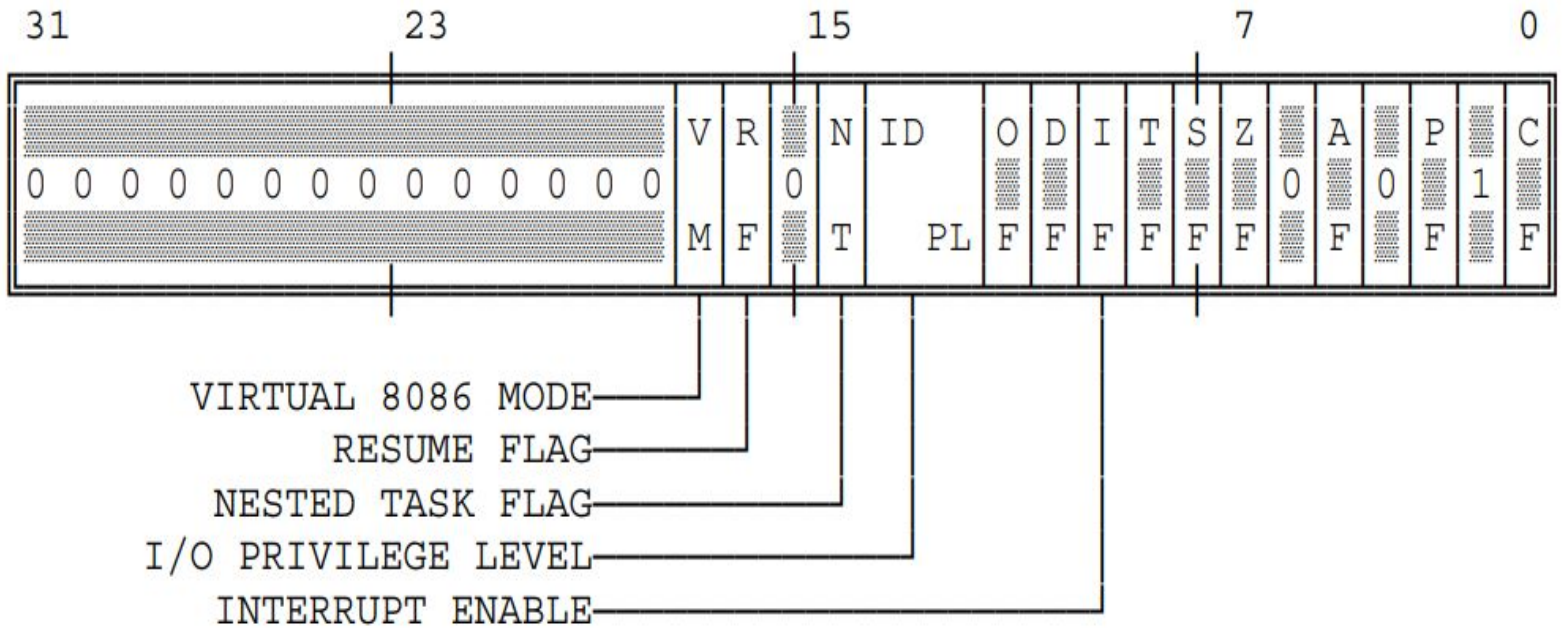
Systems Registers

- EFLAGS
- Memory Management Registers
- Control Registers
- Debug Registers
- Test Registers

Systems Registers

- ✔ **EFLAGS**
- Memory Management Registers
- Control Registers
- Debug Registers
- Test Registers

EFLAGS



NOTE

0 OR 1 INDICATES INTEL RESERVED. DO NOT DEFINE.

System Flags of EFLAG REGISTER

Systems Registers

- EFLAGS
- ✔ **Memory Management Registers**
- Control Registers
- Debug Registers
- Test Registers

Memory-Management Registers

- Four registers of the 80386 locate the data structures that control segmented memory management:
  - ✔ GDTR Global Descriptor Table Register
  - ✔ LDTR Local Descriptor Table Register

- These registers point to the segment descriptor tables GDT and LDT.

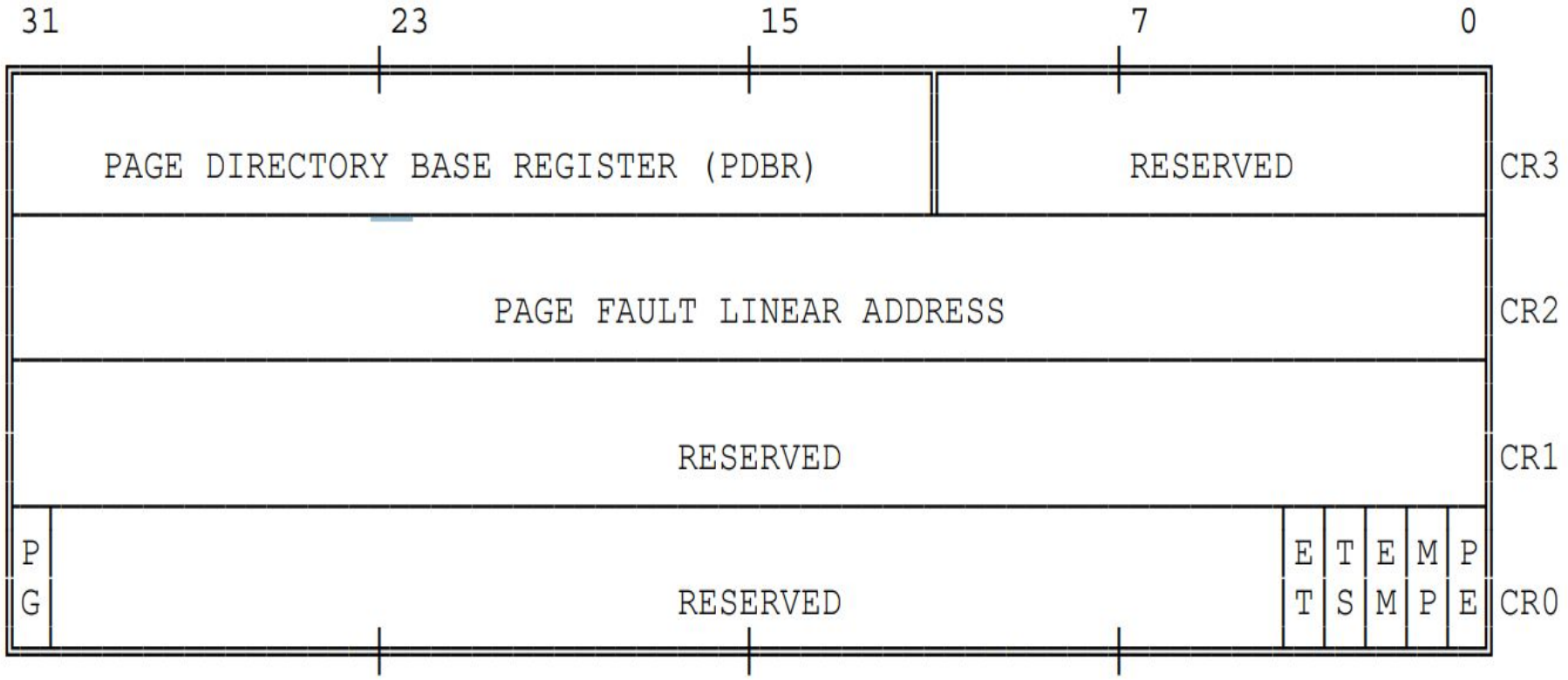
- IDTR Interrupt Descriptor Table Register
  This register points to a table of entry points for interrupt handlers (the IDT).

- TR Task Register
  This register points to the information needed by the processor to define the current task.

Systems Registers

- EFLAGS
- Memory Management Registers
- ✔ **Control Registers**
- Debug Registers
- Test Registers

Control Registers



Systems Registers

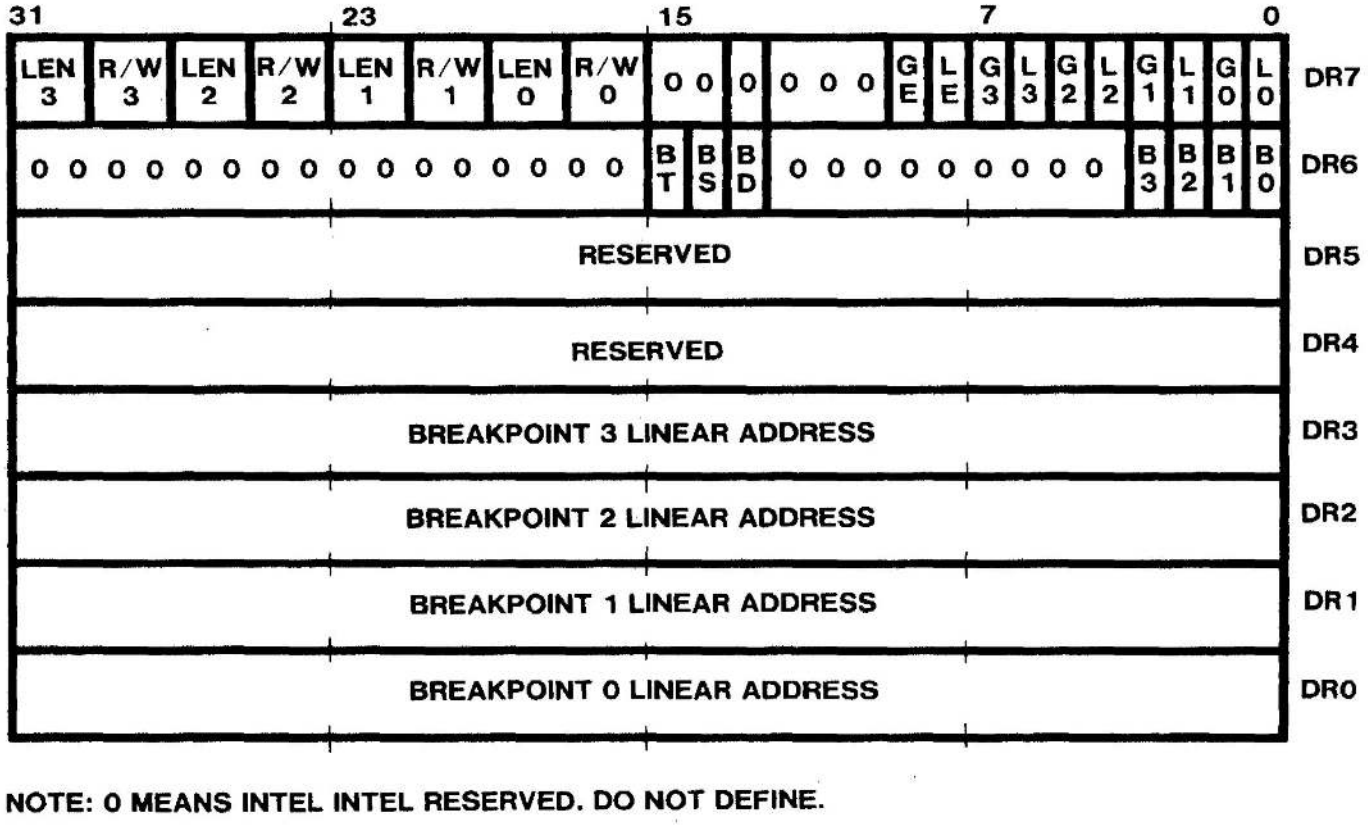
- EFLAGS
- Memory Management Registers
- Control Registers
- ✔ **Debug Registers**
- Test Registers

Debug Registers

- Six registers: to control debug features
- Accessed by variants of the MOV instruction
- debug registers are privileged resources

Registers are:

- ✔ Debug Address Registers (DRO-DR3)
- ✔ Debug Status Register (DR6)
- ✔ Debug Control Register (DR7)



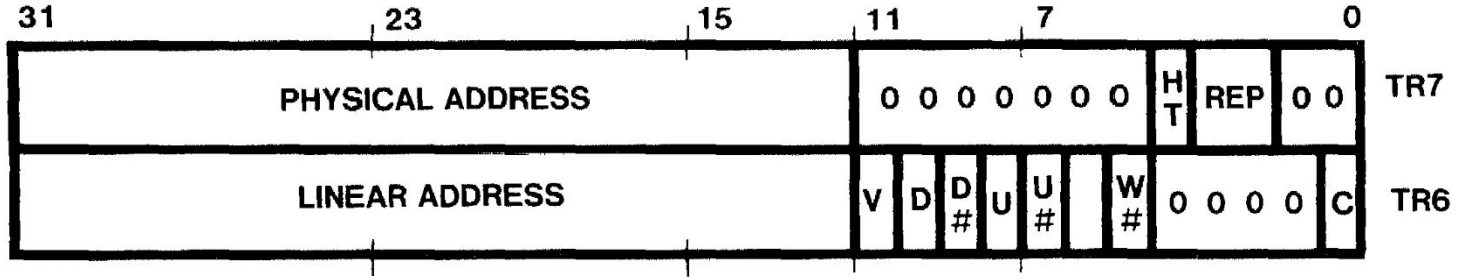
Systems Registers

- EFLAGS
- Memory Management Registers
- Control Registers
- Debug Registers

✔ **Test Registers**

Test Registers

- Two test registers are provided for the purpose of testing.
- TR6 is the test command register, and TR7 is the test data register.
- The test registers are privileged resources; in protected mode, the MOV instructions that access them can only be executed at privilege level 0



The Test Command Register (TR6)

- C: Command bit, two commands: '0'- write entries into the TLB and '1' perform TLB lookups

- Linear Address:
  - ✔ on a TLB write command, a TLB entry is allocated to this linear address and the rest of that TLB entry is set as per the values of TR7 & TR6
  - ✔ on a TLB lookup command, the TLB is interrogated as per this value and if one and only one TLB entry matches, the rest of the fields of TR6 & TR7 are set from the matching TLB entry.

- V: The Valid bit for this TLB entry. The TLB uses the valid bit to identify entries that contain valid data. Entries of the TLB that have not been assigned values have zero in the valid bit. All valid bits can be cleared by writing to CR3.

- D, D#: The dirty bit for/from the TLB entry
- U, U#: The U/S bit for/from the TLB entry
- W, W#: The R/W bit for/from the TLB entry

Meaning of D, U, and W Bit Pairs

X	X#	Effect during TLB Lookup	Value of bit X after TLB Write
0	0	(undefined)	(undefined)
0	1	Match if X=0	Bit X becomes 0
1	0	Match if X=1	Bit X becomes 1
1	1	(undefined)	(undefined)

The Test Data Register (TR7)

Holds data read from or data to be written to the TLB

- **Physical Address:** This is the data field of the TLB. On a write to the TLB, the TLB entry allocated to the linear address in TR6 is set to this value. On a TLB lookup, if HT is set, the data field (physical address) from the TLB is read out to this field. If HT is not set, this field is undefined.

- **HT:** For a TLB lookup, the HT bit indicates whether the lookup was a hit (HT <- 1) or a miss (HT <- 0). For a TLB write, HT must be set to 1.

- **REP:** For a TLB write, selects which of four associative blocks of the TLB is to be written. For a TLB read, if HT is set, REP reports in which of the four associative blocks the tag was found; if HT is not set, REP is undefined.

Test Operations

✔ **To write a TLB entry**

- Move a doubleword to TR7 that contains the desired physical address, HT, and REP values. HT must contain 1. REP must point to the associative block in which to place the entry
- Move a doubleword to TR6 that contains the appropriate linear address, and values for V, D, U, and W. Be sure C=0 for "write" command.

✔ **To look up (read) a TLB entry**

- Move a doubleword to TR6 that contains the appropriate linear address and attributes. Be sure C = 1 for "lookup" command
- Store TR 7. If the HT bit in TR 7 indicates a hit, then the other values reveal the TLB contents. If HT indicates a miss, then the other values in TR 7 are indeterminate

Systems Instructions

1. **Verification of pointer parameters:**
   - ARPL —— Adjust RPL
   - LAR —— Load Access Rights
   - LSL —— Load Segment Limit
   - VERR —— Verify for Reading
   - VERW —— Verify for Writing

2. **Addressing descriptor tables:**
   - LLDT —— Load LDT Register
   - SLDT —— Store LDT Register
   - LGDT —— Load GDT Register
   - SGDT —— Store GDT Register

3. **Multitasking:**
   - LTR —— Load Task Register
   - STR —— Store Task Register

4. **Coprocessing and Multiprocessing :**
   - CLTS —— Clear Task-Switched Flag
   - ESC —— Escape instructions
   - WAIT —— Wait until Coprocessor not Busy
   - LOCK —— Assert Bus-Lock Signal

Systems Instructions

5. Input and Output :

- IN —— Input
- OUT —— Output INS —— Input String
- OUTS —— Output String

6. Interrupt control :

- CLI —— Clear Interrupt-Enable Flag
- STI —— Set Interrupt-Enable Flag
- LIDT —— Load IDT Register
- SIDT —— Store IDT Register

7. Debugging :

- MOV —— Move to and from debug registers

8. TLB testing :

- MOV —— Move to and from test registers

9. System Control:

- SMSW —— Set MSW
- LMSW —— Load MSW
- HLT —— Halt Processor
- MOV —— Move to and from control registers


Thank you

Thank you


CONCLUSION SLIDES
Enter your sub headline here
THANK
YOU


Que And Ans