UNIT 1

80386 DX Basic Programming Model and Application Instruction Set
**FEATURES OF 80386:**

Two versions of 80386 are commonly available:

1) 80386DX
2) 80386SX

<table>
<thead>
<tr>
<th>Feature</th>
<th>80386DX</th>
<th>80386SX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) 32 bit address bus</td>
<td>32 bit address bus</td>
<td>24 bit address bus</td>
</tr>
<tr>
<td>32 bit data bus</td>
<td>32 bit data bus</td>
<td>16 bit data bus</td>
</tr>
<tr>
<td>2) Packaged in 132 pin ceramic</td>
<td>Packaged in 132 pin ceramic (PGA)</td>
<td>100 pin flat package</td>
</tr>
<tr>
<td>Address 4GB of memory</td>
<td>Address 4GB of memory</td>
<td>16 MB of memory</td>
</tr>
<tr>
<td>3) 16 MB of memory</td>
<td>16 MB of memory</td>
<td>16 MB of memory</td>
</tr>
</tbody>
</table>
Features of 80386

- 32 bit processor, it has 32 bit ALU which allows to process 32 bit data at a time.
- 32 bit address bus, therefore it can access 4GB physical memory and 64 Terabytes of Virtual memory.
- It has pipeline architecture which allows simultaneous instruction fetching, decoding, and executing and memory management. Because of instruction pipelining higher bus bandwidth & on chip address translation mechanism, the average execution time has been significantly reduced.
- It allows user to switch between different OS such as DOS and UNIX
- Operates in Real, Protected and Virtual 8086 mode.
Features of 80386

- It compatible with 8086, 8088, 80186, 80286 architecture.
- It has different data types like bits, byte, word, double word, Quadword, Tenbytes integer (signed and unsigned form).
- It has Separate pins for its address and data line, this result in higher performance and easier hardware design.
- Prefetch unit permits to prefetch upto 16bytes of instruction code. Therefore fetch time for most instruction is hidden, increase the performance.
Features of 80386

- It contains dedicated hardware for performing high speed address calculation, logical to linear address translation and protection check.

- It contain translation lookaside buffer (TLB) that store recently used page directory and page table entries. This buffer consists of 32 sets of table entries which allow direct access of 128kbyts of paged memory.
Data Types

- Fundamental Data Types
Architecture of 80386
The Internal Architecture of 80386 is divided into 3 sections.

1 Central processing unit (CPU)
   a. Execution Unit
   b. Instruction Unit
2 Memory management unit (MMU)
   a. Segmentation Unit
   b. Paging Unit
3 Bus interface unit (BIU)

Central processing unit is further divided into Execution unit (EU) and Instruction unit (IU)
Architecture of 80386

1 Central processing unit (CPU)

a) **Execution Unit**: Reads the instruction from the instruction queue and execute the instruction. Consists of three subunits: control, data and protection test unit.

i) **Control Unit**: It contains microcode and special hardware allows processor to reduce time required for execution of multiply and divide instruction. It also speeds the effective address calculation.

ii) **Data Unit**: Responsible for data operations requested by the control unit. It contains ALU, eight 32 bit general purpose registers and 64 bit barrel shifter. The barrel shifter is used for multiple bit shifts in one clock. Thus it increases the speed of all shift and rotate operations.
Architecture of 80386

III) Protection Test Unit: checks for segmentation violations under the control of the microcode.

b) Instruction Decode Unit: Takes instruction byte from the code prefetch queue and translates them into microcode. The decoded instructions are then stored in the instruction queue.

2) Memory Management Unit

a) Segmentation Unit: Translate logical address to linear addresses at the request of execution unit. Compares the effective address for the length limit specified in the segment descriptor. Adds the segment base and the effective address to generate linear address. Before calculation of linear address it also checks access rights. It provides a 4 level protection mechanism for protecting and isolating the system code and data from those of the application program.
b) Paging Unit: Translate linear addresses generated by the segmentation unit into physical addresses. If paging unit is not enabled, the physical address is the same as the linear address. It give physical address to the Bus Interface Unit to perform memory and I/O accesses. It organizes the physical memory in terms of pages of 4kbyts size each.

3) Bus Control Unit:

Is the 80836 communication with the outside world. It provides a full 32 bit bi-directional data and 32-bit address bus. Responsible for following operations

i) Accepts internal requests for code fetch and for data transfers from the code fetch unit and from the execution unit. It then prioritize the request and generates signals to perform bus cycles.
Architecture of 80386

ii) It send address, data and control signals to communicate with memory and I/O devices

iii) It controls the interface to external bus masters and coprocessors.

iv) It also provides the address relocation facility.

Instruction Prefetch Unit

Fetches sequentially the instruction byte stream from the memory. It uses bus control unit to fetch instruction bytes when it is not performing bus cycles. These prefetched instruction bytes are stored in 16 bytes code queue. When jump or call instructions are executed, the contents of the prefetched and decode queues are cleared out
Instruction Predecode Unit:

Takes instruction byte from the instruction perfetch queue and translate them into microcode. The decoded instruction then stored in instruction queue.
Register
Organization
Register Organization

- General Purpose Register
- Pointer register
- Index register
- Segment Register
- Eflags
- System Address/Memory management Registers
- Control Register
- Debug Register
- Test Register
Register Organization

- **General Purpose Register**
  - The 32-bit general-purpose registers (EAX, EBX, ECX, EDX).
  - The 16-bit general-purpose registers (AX, BX, CX, DX).
  - The 8-bit general-purpose registers (AH, BH, CH, DH, AL, BL, CL, or DL).

- **Pointer Register**
  - The 32 bit pointer register (EBP, ESP)
  - The 16-bit Pointer registers (BP, SP)
Register Organization

- **Index Register**
  - The 32 bit Index register (ESI, EDI)
  - The 16-bit Pointer registers (SI, DI)

- **Instruction Pointer**
  - The 32 bit Instruction pointer (EIP)

- **Segment Registers**
  - The **16-bit** segment registers (CS, DS, SS, ES, FS, and GS).
General Purpose Register

Segment Register

Index Register

Pointer Register
**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.
Flag Register

- Six Conditional Flags
  - Carry Flag (CF)
  - Parity Flag (PF)
  - Auxiliary Flag (AF)
  - Zero Flag (ZF)
  - Sign Flag (SF)
  - Overflow Flag (OF)

- Three Control Flags
  - Interrupt Flag (IF)
  - Trap Flag (TF)
  - Direction Flag (DF)
Flag Register

- Four System Flags
  - Input/output privilege level (IOPL)
  - Nested Task (NT)
  - Resume Flag (RF)
  - Virtual Mode Flag (VM)
- **VM - Virtual Mode Flag:**
  - Indicates operating mode of 80386.
  - When VM flag is set, 80386 switches from protected mode to virtual 8086 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.
- **NT (Nested flag):**
  - This flag is set when one system task invokes another task. (i.e. nested task).

- **IOPL (I/O Privilege level):**
  - The two bits in the IOPL are used by the processor and the operating system to determine your application's access to I/O facilities.
System Address Registers

The 386 supports four types of descriptor table:

- Global descriptor table (GDT),
- Local descriptor table (LDT),
- Interrupt descriptor table (IDT), and
- Task state segment descriptor (TSS).

Four special registers are defined to hold the base address of these tables:

- Global descriptor table Register (GDTR),
- Local descriptor table Register (LDTR),
- Interrupt descriptor table Register (IDTR), and
- Task state segment descriptor Register (TR).
Control Register (32-bit)

Figure 4-2. Control Registers

- **CR3**: 31 23 15 7 0
  - **PAGE DIRECTORY BASE REGISTER (PDBR)**
  - **RESERVED**

- **CR2**: 31 23 15 7 0
  - **PAGE FAULT LINEAR ADDRESS**

- **CR1**: 31 23 15 7 0
  - **RESERVED**

- **CR0**: 31 23 15 7 0
  - **RESERVED**
    - **PTE**: Page Table Entry
    - **PE**: Present Flag
  - **SE**: System Execution Disable Flag
  - **EM**: Execution Restrictions Mode
  - **MP**: Memory Protection
  - **PE**: Present Flag
Control Register 0 (CR0)

CR0 contains system control flags, which control or indicate conditions that apply to the system as a whole, not to an individual task (Also know as MSW).

- **PE (Protection Enable bit 0)**
  - Setting PE causes the processor to begin executing in protected mode. Resetting PE returns to real-address mode (PE=1 System in protected mode else in real mode)

- **MP (Math Present, bit 1)**
  - The MP (monitor coprocessor) bit indicates whether a coprocessor is actually attached

- **EM (Emulation, bit 2)**
  - EM=1 cause type 7 interrupt for each ESC instruction. ESC instruction is used to encode instruction for 80387.
Control Register 0 (CR0)

- **TS (Task Switch, bit 3)**
  - Indicate task has been switched to 80387 (TS=1 numeric coprocessor cause type7 interrupt).

- **ET (Extension Type, bit 4)**
  - ET indicates the type of coprocessor present in the system (80287 or 80387 ). (ET=0 select 80287; ET=1 select 80387)

- **PG(Paging , bit 31)**
  - PG indicates whether the processor uses page tables to translate linear
Control Register

- **Control Register 1 (CR1).**
  - reserved by Intel

- **Control Register 2 (CR2)**
  - Read only register. The 80386 itself writes the last 32 bit linear address of page fault routine in this register. When page fault occurs, processor generates exception 14 (page fault).
  - This address is important for writing page fault routine.

- **Control Register 3 (CR3)**
  - Control register 3 hold the base address of page directory
  - It also called Page Directory Base Register (PDBR)
Debug Register

- 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.

- Breakpoint address may locate an instruction or datum.
### Debug Register

#### Figure 12-1. Debug Registers

<table>
<thead>
<tr>
<th>Length</th>
<th>R/W</th>
<th>31</th>
<th>23</th>
<th>15</th>
<th>7</th>
<th>0</th>
<th>DR7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Notes

- **DR0**
- **DR1**
- **DR2**
- **DR3**
- **DR4**
- **DR5**
- **DR6**
- **DR7**

- **BREAKPOINT 0 LINEAR ADDRESS**
- **BREAKPOINT 1 LINEAR ADDRESS**
- **BREAKPOINT 2 LINEAR ADDRESS**
- **BREAKPOINT 3 LINEAR ADDRESS**

- **RESERVED**
- **RESERVED**

- **NOTE**
  - Ø MEANS INTEL RESERVED. DO NOT DEFINE.
Debug Address Register (DR0-DR3)

- Each of these registers contains the linear address associated with one of four breakpoint conditions. Each breakpoint condition is further defined by bits in DR7.
- The debug address registers are effective whether or not paging is enabled. The addresses in these registers are linear addresses.
Debug Status Register (DR7)

- The debug status register permits the debugger to determine which debug conditions have occurred.

- For each address in registers DR0-DR3, the corresponding fields R/W0 through R/W3 specify the type of action that should cause a breakpoint. The processor interprets these bits as follows:
  - 00 -- Break on instruction execution only
  - 01 -- Break on data writes only
  - 10 -- undefined
  - 11 -- Break on data reads or writes but not instruction fetches
Debug Status Register (DR7)

- The LE and GE bits control the "exact data breakpoint match" feature of the processor. If either LE or GE is set, the processor slows execution so that data breakpoints are reported on the instruction that causes them. It is recommended that one of these bits be set whenever data breakpoints are armed. The processor clears LE at a task switch but does not clear GE.
Real, protected and virtual mode

Segmentation = memory management scheme

Segment = block of memory.

A logical address consists of:

• base address

• offset (relative address within the segment)
Real, protected and virtual mode

- Advantages of the segmentation
  - shorter addressing part of the instruction (only the offset of the operand is encoded in the instruction, the base address is in a base register)
  - instructions are separated from data

- Segment types
  - code segment – holds machine instructions
  - data segment
  - stack segment – holds: o return addresses, parameters and local variables of procedures, temporary results of mathematical operations
Real mode

Processor 8086 – works only in real mode.

20-bit address bus => memory ??

Offset ... 16-bit value =>

- base address
- offset
- linear address
Protected mode

32-bit address bus => memory??

Offset ... 16-bit or 32-bit value

Linear address = base address + offset
Virtual mode

- Processor runs in protected mode, but simulates real mode: a 20-bit linear address is translated by paging to a 32-bit physical address.

- A processor is switched to virtual mode when running a DOS application under Windows operating system.
MEMORY
MANAGEMENT
Memory Management

- The 80386 transforms logical addresses (i.e., addresses as viewed by programmers) into physical address (i.e., actual addresses in physical memory) in two steps:

  - **Segment translation**, in which a logical address (consisting of a segment selector and segment offset) are converted to a linear address.

  - **Page translation**, in which a linear address is converted to a physical address. This step is optional, at the discretion of systems-software designers.
Memory Management
Segment Translation

To perform this translation, the processor uses the following data structures:

- Descriptors
- Descriptor tables
- Selectors
- Segment Registers
Segment Translation
Segment Translation

- Segment register contains a selector that selects a descriptor from the descriptor table.
- The descriptor contains information about the segment, e.g., it's base address, length and access right
- The offset can be 32-bits.
A segment selector is loaded into a segment register (cs, ds, etc.) to select one of the regular segments in the system as the one addressed via that segment register.
Segment Descriptor – 64 bit
Segment Descriptor

- **Base Address**: Starting address of the memory segment
- **Limit**:
  - Length of the segment minus 1.
  - 20-bits allows segments up to 1 MB.
  - This value is shifted by 12 bits to the left when the G (Granularity bit) is set to 1.
- **G (Granularity) Bit**: When G=0, segments can be 1 byte to 1MB in length. When G=1, segments can be 4KB to 4GB in length.
Segment Descriptor

- **U bit:**
  User (OS) defined bit.

- **X Bit:**
  Reserved by Intel

- **D bit:**
  Indicates how the instructions (80386 and up) access register and memory data in protected mode.
  - When D=0, instructions are 16-bit instructions, with 16-bit offsets and 16-bit registers. Stacks are assumed 16-bit wide and SP is used.
  - When D=1, 32-bits are assumed. Allows 8086-80286 programs to run
Segment Descriptor (Access Right byte)

A = 0, Segment not accessed
A = 1, Segment has been accessed

S = 0, System descriptor
S = 1, Code, data or stack

Sets the desc. privilege level.

P = 0, descriptor is undefined.
P = 1, descriptor contains a valid base and limit.

The Access Rights (AR) byte controls access to a protected mode segment and how the segment functions in the system.

000  Data, read-only
001  Data, read/write
010  Stack, read-only
011  Stack, read/write
100  Code, execute-only
101  Code, execute/read
110  Code, execute-only, conforming
111  Code, execute/read, conforming
Types of Segment Descriptor

- System
  - LDT
  - TSS
  - Gate

- Non-system
  - Code
  - Stack
  - Data
Non System Descriptor

S=1

Non System

A=0, Segment not accessed
A=1, Segment has been accessed

000  Data, read-only
001  Data, read/write
010  Stack, read-only
011  Stack, read/write
100  Code, execute-only
101  Code, execute/read
110  Code, execute-only, conforming
111  Code, execute/read, conforming
**System Descriptor**

A=0, Segment not accessed  
A=1, Segment has been accessed

<table>
<thead>
<tr>
<th>Type</th>
<th>Defines</th>
<th>Type</th>
<th>Defines</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved by Intel</td>
<td>8</td>
<td>Reserved by Intel</td>
</tr>
<tr>
<td>1</td>
<td>Available 80286 TSS</td>
<td>9</td>
<td>Available Intel 80286 TSS</td>
</tr>
<tr>
<td>2</td>
<td>LDT</td>
<td>A</td>
<td>Undefined</td>
</tr>
<tr>
<td>3</td>
<td>Busy 80286 TSS</td>
<td>B</td>
<td>Busy Intel 80386DX</td>
</tr>
<tr>
<td>4</td>
<td>80286 Call Gate</td>
<td>C</td>
<td>Intel 80386DX Call gate</td>
</tr>
<tr>
<td>5</td>
<td>Task Gate</td>
<td>D</td>
<td>Undefined</td>
</tr>
<tr>
<td>6</td>
<td>80286 Interrupt Gate</td>
<td>E</td>
<td>80386DX Interrupt Gate</td>
</tr>
<tr>
<td>7</td>
<td>80286 Trap Gate</td>
<td>F</td>
<td>80386DX Trap Gate</td>
</tr>
</tbody>
</table>
Descriptor Tables

➢ U bit: