The Intel 8086 Architecture and Programming

Topics:

1. Registers and internal architecture (Ch 2)

2. Address generation (Ch 2)

3. Addressing modes (Ch 3)

4. Assembly and Machine Language (Ch 4)

5. 8086/88 Hardware specifications and bus structure (Ch 8)

6. SDK 86 System Development Kit (Lab)

Registers and Internal Architecture

There are two main functional logic block in the 8086/88 processors:

**EU** Execution Unit - execution of program instructions

**BIU** Bus Interface Unit - provides interface to memory and I/O

1. controls the address, data and control busses.

2. handles instruction fetch and data read/write functions
Internal Architecture of the 8086/88

Execution Unit

- General Registers
- Temporary Registers
- Flag Register

Bus Interface Unit

- Address Bus
- Data Bus
- Instruction Queue

ALU

Bus Control Logic

8086

- CPU
- COP
- 80486DX
- 8K Cache

80286

- CPU
- COP
- Pentium
- 16K Cache

80386DX

- CPU
- COP
- Pentium Pro
- int
- int
- fp
- 16K L1 cache
- 256K L2 cache
- RISC cores

Pentium III

Pentium IV
EAX Accumulator: used for arithmetic and logic operations. Destination for MUL and DIV.

EBX Base Index: Typically used to hold offset addresses.

ECX Count: Typically used to hold a count value for various instructions (repeated strings, LOOP/LOOPD, Shift/rotate).

```
MOV CX, 080H
HERE . . . . . ;
LOOP HERE ; Decrement CX, JNZ HERE
```

EDX Data: temporary data storage for part of a result from a multiplication (Most significant result) or division (dividend, remainder).
ESP Stack Pointer: Used to offset into the stack segment to address the stack. PUSH/POP JSR

EBP Base Pointer: Used to store a base memory location for data transfers.

EDI Destination Index: Typically used as an offset for the destination memory location for string/byte transfers.

ESI Source Index: Typically used as an offset for the source memory location for string/byte transfers.

The use of the base and offset registers EBX, ESP, EFP, EDI and ESI will become clearer when addressing modes are covered.

---

Flag Register (Execution Unit)

<table>
<thead>
<tr>
<th>EFLAGS</th>
<th>FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30</td>
<td>16 15 14 13 12 11 10</td>
</tr>
</tbody>
</table>

Note: O,Z,A,P & C are changed by most arithmetic and logic instructions but are unchanged by data transfers.

C Carry: Holds the carry after addition, or the borrow after subtraction.

P Parity: ‘0’ - odd parity. ‘1’ - even parity.

A Auxiliary carry: Holds the “half-carry/borrow” after addition/subtraction. (BCD operations on nibbles).

Z Zero: ‘1’ if the result of an arithmetic or logic operation is zero.
**S** Sign: holds the sign of the result after a arithmetic or logical operation. This is the value of the sign bit of the result of the operation.

**T** Trap: enables trapping if '1'. Program flow is interrupted based on the values of the control and debug registers.

**I** Interrupt: Controls the operation of the INTR (interrupt request) pin. If '1', interrupts from INTR are enabled.

**D** Direction: Selects either increment or decrement for the SI and/or DI registers during string and loop functions. If '1', the registers are decremented.

**O** Overflow: Indicates that a result has exceeded the capacity of a register during *signed* operations.

**IOP** (80286) I/O Privilege level: Two bits correspond to privilege level for I/O operation. 00 is the highest, 11 is the lowest.

**NT** (80286) Nested task: Set when a task is nested within another task.

**RF** (80386) Resume Flag: Used during debugging.

**VM** (80386) Virtual Mode: Virtual mode execution (multiple 8086s running in protected mode).

**AC** (80486SX) Alignment Check: Non-aligned address (for co-processor).
**VIF** (Pentium) Virtual Interrupt Flag: A copy of the interrupt flag.

**VIP** (Pentium) Virtual Interrupt Pending:

**ID** (Pentium) ID: The CPUID instruction is supported.
**CS** Code Segment: Used to compute the starting address of the section of memory holding code (restricted to 64K in REAL mode).

**DS** Data Segment: Used to compute the starting address of the section of memory holding data (restricted to 64K in REAL mode).

**SS** Stack Segment: Used to compute the starting address of the section of memory holding the stack (restricted to 64K in REAL mode).

**ES** Extra Segment: Additional data segment used by some string instructions.

**FS&GS** Additional segment registers in the 80386 (and up) for program use.

---

**EIP** (Instruction Pointer)

- addresses the next instruction in the code segment which is to be fetched.

- Can be modified with a JMP or CALL instruction.
Address Generation

Two types of address generation:

1. Real Mode (the 8086/8088/186 can only operate in this mode)
   - Allows the μP to address the first 1Mbyte of memory only.
   - The first Meg or memory is called real or conventional memory.

2. Protected mode (80286...)
   - This mode uses the segment register contents (called a selector) to access a descriptor from a descriptor table.
   - The descriptor describes the memory segment’s location, length and access rights.

Real Mode Memory Addressing

Memory addresses consist of a segment address plus and offset address.

- The segment address defines the start of a 64K block of memory.

- The offset address selects a location within the 64K memory segment.

- Memory locations are often written as:

  segment:offset
  C000:04BA
Address Generation

The µP has a set of rules that apply whenever memory is addressed, which define the segment and offset register combination used by certain addressing modes.

<table>
<thead>
<tr>
<th>Segment</th>
<th>Offset</th>
<th>Special Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>IP</td>
<td>Instruction address</td>
</tr>
<tr>
<td>SS</td>
<td>SP or BP</td>
<td>Stack address</td>
</tr>
<tr>
<td>DS</td>
<td>BX,DI,SI</td>
<td>Data address</td>
</tr>
<tr>
<td>ES</td>
<td>DI</td>
<td>String destination</td>
</tr>
</tbody>
</table>

Notes:

1. Memory segments may overlap.

2. The segment-offset scheme allows programs to be relocated in memory (on 16 byte boundaries). Move the existing contents to the new physical location, then update the segment register.