3. Programming Model
Data Registers

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AX</td>
<td>AH</td>
<td>AL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BX</td>
<td>BH</td>
<td>BL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CX</td>
<td>CH</td>
<td>CL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DX</td>
<td>DH</td>
<td>DL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Four 16-bit general purpose registers used to store operands and results.

• Each can be accessed as a whole (AX, BX, CX & DX), or the upper and lower bytes can be accessed separately (AL, AH, BL, BH, CL, CH, DL & DH).

• AX is an implied operand for some arithmetic operations.

• BX may be used as a base register in address calculations.

• CX is an implied counter for certain instructions.

• DX is sometimes used to hold the address of I/O instructions.
Pointer and Index Registers

<table>
<thead>
<tr>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td></td>
</tr>
<tr>
<td>BP</td>
<td></td>
</tr>
<tr>
<td>SI</td>
<td></td>
</tr>
<tr>
<td>DI</td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td></td>
</tr>
</tbody>
</table>

- IP is the Instruction Pointer, essentially the program counter.
- SP is the Stack Pointer.
- BP is a base register for accessing the stack. Very often used with other registers and/or a displacement.
- SI and DI are used for indexing very often used in combination with BX or BP and/or a displacement.
Segment Registers

<table>
<thead>
<tr>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td></td>
</tr>
<tr>
<td>DS</td>
<td></td>
</tr>
<tr>
<td>ES</td>
<td></td>
</tr>
</tbody>
</table>

- Used to calculate physical memory addresses.
- CS (Code Segment) is used in combination with IP to access the executing program instructions.
- SS (Stack Segment) is used in combination with SP to access the executing program Stack.
- DS (Data Segment) and ES (Extra Segment) are used to access the executing program data.
Physical Address

- The Segment, Pointer and other addressing registers are only 16 bits wide. On the other hand, the address put on the bus by the 8086 (called Physical Address) is 20-bits. So, where do the missing 4-bits come from?

\[
\begin{array}{c}
\text{Effective Address or Offset} \\ 16 \text{ Bits}
\end{array} + 
\begin{array}{c}
\text{Segment Address} \\ 16 \text{ Bits}
\end{array} = 
\begin{array}{c}
\text{Physical Address} \\ 20 \text{ Bits}
\end{array}
\]

Example: \((CS) = 123A, (IP) = 341B\)

Therefore, next instruction fetched at:

\[
\begin{align*}
341B & \quad \text{Offset} \\
+ \quad 123A0 & \quad \text{Segment Address} \\
= \quad 157BB & \quad \text{Physical Address}
\end{align*}
\]
The 8086 PSW contains 16 bits: 9 Flags & 7 not used.

• Flags are divided into *conditional* and *control*.

• Conditional Flags reflect the result of the previous operation involving the ALU.

• Control Flags control the execution of special functions or instructions.

• SF (Sign Flag) is equal to the most significant bit (MSB) of the result (2’s complement numbers has a “1” as MSB).

• ZF (Zero Flag) is ‘1’ if the result is zero.

• PF (Parity Flag) is ‘1’ if the low order 8-bits of the result contain an even number of ‘1s’.

• CF (Carry Flag) is ‘1’ if an addition causes a carry-out or a subtraction causes a borrow.
• AF (Auxiliary Flag) is ‘1’ if an addition causes a carry-out or a subtraction causes a borrow in/by bit 3. This flag is used exclusively for BCD arithmetic.

• OF (Overflow Flag) is set if the result is out of range.

• DF (Direction Flag) is used by string manipulation instructions. If ‘1’ auto-increment; otherwise, auto-decrement.

• IF (Interrupt Flag) if set, maskable interrupts are recognized.

• TF (Trap Flag) if set, a “trap” is executed after each instruction (control is transferred to a subroutine after each instruction).
Program Memory Models

- Tiny - single segment shared for code, data & stack (CS = DS = SS)
- Small - single segment for code, separate single segment shared for data & stack (CS ≠ DS = SS)
- Medium – multiple segments for code, separate single segment shared for data & stack (CS ≠ DS = SS)
- Compact - single segment for code, separate multiple segment for data & stack (CS ≠ DS ≠ ES ≠ SS)
- Large - multiple segments for Code, Data, Stack (CS ≠ DS ≠ ES ≠ SS)

Open question: How do you decide which one to use?
Programming Model Exercise

Fill in the blanks/answer the questions:

1. Given CS = 2000H, SS = F000H, IP = 1234H & SP = 5558H, what address will the CPU put on the bus if accessing the Stack? ______

2. _____ is used as an implied counter.

3. _____ is sometimes used to hold I/O addresses.

4. Program Counter: _____

5. Base register to access the Stack: _____

6. _____ & _____ are used to compute the address of the next instruction to be fetched.

7. If AX = 60FFH and BX = 9F01H, after adding the contents of AX & BX storing the result on AX, what is the status of the following flags?

   SF = ___   ZF = ___   PF = ___
   CF = ___   OF = ___

8. Suppose that PSW = 0ADC5H, will a hardware interrupt occur if the CPU INTR pin is asserted? Will it occur if the NMI pin is asserted?