Processor Registers

Flags
AX
BX
CX
DX
SI
DI
BP
SP
IP
CS
DS
ES
SS

Status Flags:
CF: carry
PF: parity
AF: auxiliary carry
ZF: zero
SF: sign
OF: overflow
IF: interrupt flag
DF: direction flag (for string instructions)
TF: trace flag (used by debugger)
IOPL: i/o privilege level (used in protected mode)
NT: nested task flag (used in protected mode)

Microprocessor Operation

Machine Cycles
Instruction Fetch
Instruction Decode
Instruction Execution

Bus Cycles
Memory Read
Memory Write
I/O Read
I/O Write
Interrupt Acknowledge

Control Signals from Bus
Interrupt
Non-Maskable Interrupt
Wait
Hold (for DMA)
Multiprocessor synchronization

**Data Types and Assembler Data Declaration Pseudo-Ops**

**Data Sizes:** Operands will have a size, which is one of the following:

- **BYTE** 8 bits
- **WORD** 16 bits, 2 BYTES
- **DWORD** (double word) 32 bits, 2 WORDS or 4 BYTES
- **QWORD** (quad word) 64 bits, 4 WORDS, 8 BYTES
- **TBYTE** 80 bits, 10 BYTES

Character data are arrays of BYTES

**Pseudo-Ops**

- **DB** - Define Byte
- **DW** - Define Word (16 bits)
- **DD** - Define Doubleword (32 bits)
- **DQ** - Define Quadword (64 bits)
- **DUP** - assembler pseudo function to replicate data
- **EQU** - define symbol

**Programmer’s Model of the Machine**

**Memory Organization:**

- 8086 processor has 20 bit address bus, can address 1 MB of memory
- 80286 has 24 bit address bus, can address 16MB of memory
- 80386 and later has 32 bit address bus, can address 4GB of memory

Memory above 1MB boundary can only be accessed in protected mode.

- Original IBM PC had memory organized as array of bytes
- PC-AT had memory organized as array of words
- 80386 and later processors have memory organized as array of dwords

- In all cases, memory is byte addressable. There are no alignment restrictions, but performance can be increased by using proper alignment.
- In all cases, memory is segmented and all addresses use both a segment and offset.
- In many cases, the segment is implied in the operation of the instruction. The segment address used by an instruction will always come from one of the cpu’s segment registers.

Real mode physical address calculation:

\[
PA = \text{seg} \ll 4 + \text{off}
\]

Interrupt vector table is 1k bytes starting at 0000:0000 (0:0-0:3FFh)
Reset vector is at FFFF:0

Little Endian vs. Big Endian – Intel processors are little-endian. Motorola processors are big-endian. The leftmost bit (bit 7, bit 15, bit 31) is considered to be the most significant bit.
**Addressing Modes**

The microprocessor instructions operate on data. The addressing mode for the instruction informs the processor how to find the data, (operands) for the instruction to operate on.

- **Implied or inherent:** The location of the operand is inherent in the instruction.  
  ex: `stc`

- **Register:** The operand is contained in one of the processor registers:  
  ex: `mov ax,bx`

- **Immediate:** The operand is included as part of the instruction.  
  ex: `mov ax,1`

- **Direct:** The address of the data is included in the instruction.  
  ex: `mov ax,var1`

- **Register Indirect:** The address of the data is specified by one of the following registers: BX, SI, DI.  
  ex: `mov ax,[bx]`

- **Indexed:** The address of the data is contained in an index register, plus an optional offset.  
  ex: `mov ax,[si+5]`

- **Based:** The address of the data is contained in a base register, plus an optional offset.  
  ex: `mov ax,[bp+5]`

- **Based Indexed:** The address of the data is contained in the sum of a base register plus an index register, plus an optional offset.  
  ex: `mov ax,[bx][di]+5`

**Summary of Memory Operand Addressing Modes**

<table>
<thead>
<tr>
<th>Addressing Mode</th>
<th>Offset Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>16 bit immediate displacement</td>
</tr>
<tr>
<td>Register Indirect</td>
<td>BX, SI, DI</td>
</tr>
<tr>
<td>Based</td>
<td>(BX or BP) + {displacement}</td>
</tr>
<tr>
<td>Indexed</td>
<td>(SI or DI) + {displacement}</td>
</tr>
<tr>
<td>Based Indexed</td>
<td>(BX or BP) + (SI or DI)</td>
</tr>
<tr>
<td>Based Indexed + Displacement</td>
<td>(BX or BP) + (DI or DI) + displacement</td>
</tr>
</tbody>
</table>

Note that the above applies to processors from the 8086 to the 80286. The 32 bit processors (80386 and above), are much less restrictive in which registers can be used as base registers and index registers.
Segment Selection for Memory Addressing:

- **Code Segment:** CS
- **Data Segment:** DS
- **Stack Segment:** SS

Implied segment usage by index, pointer and base registers

- **SP:** SS
- **BP:** SS
- **BX:** DS
- **SI:** DS
- **DI:** DS (or ES for some string instructions)

Based addressing using BX uses DS. ([BX]+offset)
Based addressing using BP uses SS ([BP]+offset)
Based Indexed addressing uses DS ([BP]+[SI]+offset)

Default segment usage may be overridden with an explicit segment override prefix byte.

Interrupts:

There are *three basic types* of interrupt.

- **Hardware Interrupts:** Generate by external hardware associated with some I/O device.
  - **NMI** – Non-maskable interrupt. Generated by signal on the NMI pin. Can’t be disabled in software. Uses vector 2.
  - **Regular interrupt:** Generated by signal on the INT pin on the processor. Vector is supplied by external interrupt controller hardware in response to interrupt acknowledge cycle by processor.
- **Software Interrupts:** These are generated by the execution of an instruction in the program.
- **Exceptions:** These generally indicate errors detected by the processor during the execution of the program.
## INTEL Defined Interrupt Vectors

<table>
<thead>
<tr>
<th>Vector</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Divide by zero</td>
</tr>
<tr>
<td>1</td>
<td>Single step (used by debuggers)</td>
</tr>
<tr>
<td>2</td>
<td>NMI interrupt</td>
</tr>
<tr>
<td>3</td>
<td>Breakpoint (used by debuggers)</td>
</tr>
<tr>
<td>4</td>
<td>INTO instruction detected overflow</td>
</tr>
<tr>
<td>5</td>
<td>BOUND instruction detected range exception</td>
</tr>
<tr>
<td>6</td>
<td>Invalid opcode</td>
</tr>
<tr>
<td>7</td>
<td>Processor extension not available</td>
</tr>
<tr>
<td>8</td>
<td>Double fault exception detected</td>
</tr>
<tr>
<td>9</td>
<td>Processor extension segment overrun interrupt</td>
</tr>
<tr>
<td>10</td>
<td>Invalid task state segment</td>
</tr>
<tr>
<td>11</td>
<td>Segment not present</td>
</tr>
<tr>
<td>12</td>
<td>Stack segment overrun</td>
</tr>
<tr>
<td>13</td>
<td>General protection fault</td>
</tr>
<tr>
<td>14</td>
<td>Page fault</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>Floating point error</td>
</tr>
<tr>
<td>17</td>
<td>Alignment check (when enabled, this generates an exception when operands are not properly aligned)</td>
</tr>
</tbody>
</table>

Exceptions 0-5 are defined in the 8086, 6-13 were added by the 80286, 14-17 were added by the 80386 and later processors.