| <b>¢</b> ‡ General Registers |             |  |  |
|------------------------------|-------------|--|--|
| EAX                          | Accumulator |  |  |
| EBX                          | Base        |  |  |
| ECX                          | Counter     |  |  |
| EDX                          | Data        |  |  |

# **♥** Pointer Registers

| ESP | Stack Pointer, "top" of the current |  |
|-----|-------------------------------------|--|
|     | stack frame (lower memory)          |  |

EBP Base Pointer, "bottom" of the current stack frame (higher memory)

EIP Instruction Pointer, pointer to the next instruction to be executed by the CPU

# Index Registers

ESI Source Index, it is used as source index for string operations

EDI Destination Index, it is used as destination index for string operations

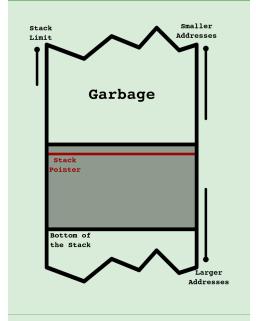
### Flags Registers

ZF Zero Flag, set when result of an operation equals zero

CF Carry Flag, set when the result of an operation is too large/small

SF Sign Flag, set when the result of an operation is negative

# ↓ Stack



Stack is a LIFO-Storage (Last In First Out)

| mov ebx, eax   | Move the value in <i>EAX</i> to <i>EBX</i>                                 |  |  |
|--|--|--|--|
| mov eax,<br>0xDEADBEEF   | Move 0xDEADBEEF into EAX   |  |  |
| mov edx,<br>DWORD PTR<br>[0x41424344]  | Move the 4-byte value at address <i>0x41424344</i> into <i>EDX</i>         |  |  |
| mov ecx,<br>DWORD PTR<br>[edx]   | Move the 4-byte value at the address in <i>EDX</i> , into <i>ECX</i>       |  |  |
| mov eax, DWORD PTR [ecx+esi*8]   | Move the value at the address <i>ECX+ESI*8</i> into <i>EAX</i>             |  |  |
| mov bx,<br>0C3EEh  | Sign bit of $BL$ is now 1:<br>$BH == 1100\ 0011$ , $BL$<br>$== 1110\ 1110$ |  |  |
| movsx ebx, bx  | Load signed 16-bit value into 32-bit register and sign-extend              |  |  |
| movzx dx, bl   | Load unsigned 8-bit value into 16-bit register and zero-extend             |  |  |
| lea edi,<br>[esi+0Bh]  | Add 11 to ESI and store the result in EDI                                  |  |  |
| <pre>eax is the value stored in eax [eax] is the value pointed to by eax</pre> |  |  |  |

| <b>≣</b> Data Types |                   |
|---------------------|-------------------|
| BYTE                | 1 Byte (8 bits)   |
| WORD                | 2 Bytes (16 bits) |
| DOUBLE WORD         | 4 Bytes (32 bits) |
| QUAD WORD           | 8 Bytes (64 bits) |

# ▶ Frequent Instructions mov MOV is the instruction used for assignment. MOV can move data between a register and memory. movsx move with Sign Extension. The data is moved from a smaller register into a bigger register, and the sign is preserved.

movzx move with Zero Extension. The data is moved from a smaller register into a bigger register, and the sign is ignored.

lea Similar to MOV, except that math can be done on the original value before it is used. The [ and ] characters always surround the second parameter, but in this case they do **not indicate dereferencing**.

push Decrements the stack pointer by the size of the operand, then saves the operand to the new address.

Equivalent to 'sub esp, 4 | mov DWORD PTR [esp], ebx'

pop Sets the operand to the value on the stack, then increments the stack pointer by the size of the operand. Equivalent to 'mov ebx, DWORD PTR [esp] | add esp, 4'

cmp Compares two operands and sets or unsets flags in the flags register based on the result.

rep, Repeat while Equal/Non Zero/Zero.
repnz,
repz