**Assembly Language**

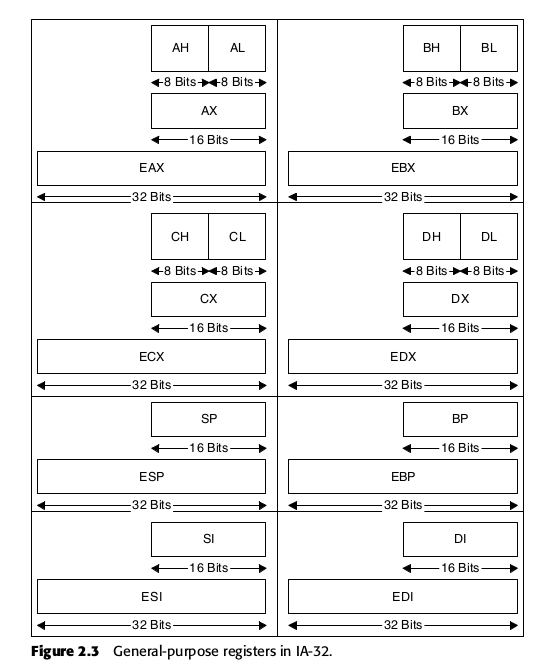
IA-32 has eight generic registers: EAX, EBX, ECX, EDX,ESI, EDI, EBP, and ESP.

Notice that all of these names start with the letter E, which stands for

extended. These register names have been carried over from the older 16-bit Intel architecture, where they had the exact same names, minus the Es (so that EAX was called AX, etc.). This is important because sometimes you’ll run into 32-bit code that references registers in that way: MOV AX, 0x1000, and so on.

**Generic IA-32 Registers and Their Descriptions**

|  |  |
| --- | --- |
| EAX, EBX, EDX | These are all generic registers that can be used for any  integer, Boolean, logical, or memory operation. |
| ECX | Generic, sometimes used as a counter by repetitive  instructions that require counting. |
| ESI/EDI | Generic, frequently used as source/destination pointers  in instructions that copy memory (SI stands for Source  Index, and DI stands for Destination Index). |
| EBP | Can be used as a generic register, but is mostly used as the stack base pointer. Using a base pointer in combination with the stack pointer creates a stack frame. A stack frame can be defined as the current function’s stack zone, which resides between the stack  pointer (ESP) and the base pointer (EBP). The base pointer usually points to the stack position right after the return address for the current function. Stack frames are  used for gaining quick and convenient access to both local variables and to the parameters passed to the current function. |
| ESP | This is the CPUs stack pointer. The stack pointer stores the current position in the stack, so that anything pushed to the stack gets pushed below this address, and this  register is updated accordingly. |



Difference between EBP and ESP : EBP points to the base of the current stack frame whereas ESP always points to the top of the stack.

What is a stack frame: A stack frame is the part allocated on the stack for a function. In a program, there can be many functions. When a new program runs, a part of stack is allocated for its main function. The main function can also call other functions, this calling allocates some stack for that function, Now that part of stack is that function ’s stack frame.

To allocate a stack frame, a process called function prologue is used, it saves the current EIP, EBP and ESP on the stack, subtracts some part from the stack and then point ESP to the new writable area. When the function finish executing, the function epilogue, releases the stack and makes it available for other functions.

IA-32 processors have a special register called EFLAGS that contains all kinds of status and system flags. The status flags, are used by the processor for recording its current logical state, and are updated by many logical and integer instructions in order to record the outcome of their actions. Additionally, there are instructions that operate based on the values of these status flags, so that it becomes possible to create sequences of instructions that perform different operations based on different input values, and so on.

**Instruction Format**

Instructions usually consist of an opcode (operation code), and one or two operands. The opcode is an instruction name such as MOV, and the operands are the “parameters” that the instruction receives (some instructions have no operands).

The general instruction format looks like this:

Instruction Name (opcode) Destination Operand, Source Operand

Some instructions only take one operand, whose purpose depends on the specific instruction. Other instructions take no operands and operate on predefined data.

