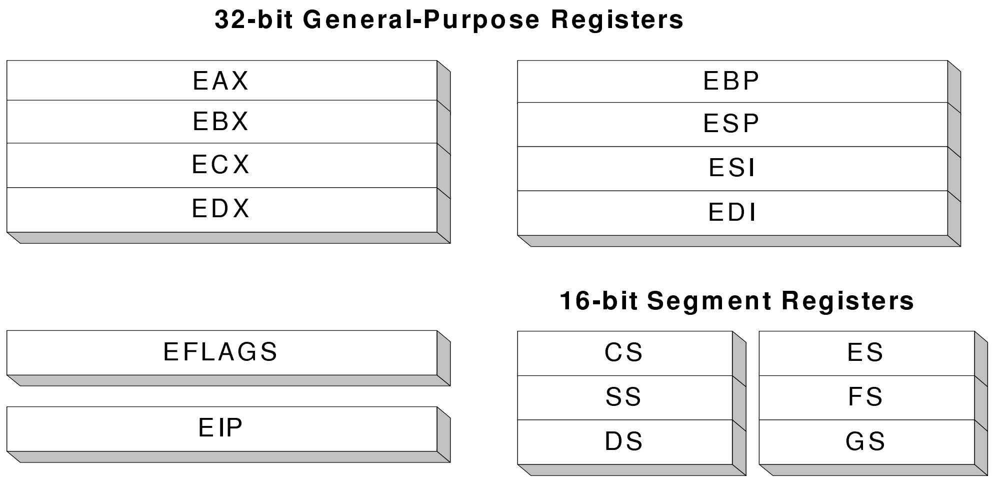
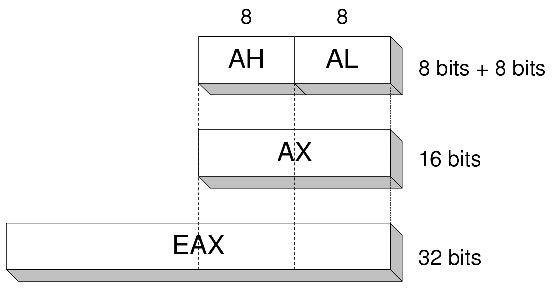
Figure 2–9 IA-32 Basic Program Execution Registers.



General-Purpose Registers The *general-purpose registers* are primarily used for arithmetic and data movement. As shown in the following figure, each register can be addressed as either a single 32-bit value or two 16-bit values:



Some registers can also be addressed as four separate 8-bit values. For example, the EAX register is 32 bits. Its lower 16 bits are also named AX. The upper 8 bits of AX are named AH, and the lower 8 bits are named AL.

This overlapping relationship exists for the EAX, EBX, ECX, and EDX registers:

32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL

32-bit	16-bit	8-bit (high)	8-bit (low)
ECX	CX	CH	CL
EDX	DX	DH	DL

The remaining general-purpose registers have separate names for their lower 16 bits, but cannot be divided further. The 16-bit registers shown here are usually used only when writing programs that run in Real-address mode:

32-bit	16-bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

Specialized Uses Some general-purpose registers have specialized uses:

- EAX is automatically used by multiplication and division instructions. It is often called the *extended accumulator* register.
- The CPU automatically uses ECX as a loop counter.
- ESP addresses data on the stack (a system memory structure). It should never be used for ordinary arithmetic or data transfer. It is often called the *extended stack pointer* register.
- ESI and EDI are used by high-speed memory transfer instructions. They are sometimes called the *extended source index* and *extended destination index* registers.
- EBP is used by high-level languages to reference function parameters and local variables on the stack. It should not be used for ordinary arithmetic or data transfer except at an advanced level of programming. It is often called the *extended frame pointer* register.

Segment Registers The *segment registers* are used as base locations for preassigned memory areas called *segments*. Some segments hold program instructions (code), others hold variables (data), and another segment called the *stack segment* holds local function variables and function parameters.

Instruction Pointer The EIP, or *instruction pointer* register contains the address of the next instruction to be executed. Certain machine instructions manipulate this address, causing the program to branch to a new location.

EFLAGS Register The EFLAGS (or just *Flags*) register consists of individual binary bits that either control the operation of the CPU or reflect the outcome of some CPU operation. There are