EE 213 Computer Organization and Assembly Language

Week # 3, Lecture # 7

30th Dhu'l-Hijjah, 1439 A.H

10th September 2018

These slides contains materials taken from various sources. I fully acknowledge all copyrights.

Minds open...

... Laptops closed

This presentation helps in delivering the lecture.

Take notes, interact and read text book to learn and gain knowledge.

Today’s Topics

- Basic Diagram of a micro-computer
- Instruction Cycle
- Simplified CPU block diagram
- Abstracted Micro-architecture of Core i7 (2013)
- Reading and Writing Memory
- Reading and Writing Input Output Devices
- Mode of Operations of x86 processors
- Address Space of x86 processors

Please read chapter # 2 (2.1 and 2.2) over the weekend.

2.2 32-Bit x86 Processors

In this section, we focus on the basic architectural features of all x86 processors. This includes members of the Intel IA-32 family as well as all 32-bit AMD processors.

2.2.1 Modes of Operation

x86 processors have three primary modes of operation: protected mode, real-address mode, and system management mode. A sub-mode, named *virtual-8086*, is a special case of protected mode. Here are short descriptions of each:

Protected Mode Protected mode is the native state of the processor, in which all instructions and features are available. Programs are given separate memory areas named *segments*, and the processor prevents programs from referencing memory outside their assigned segments.

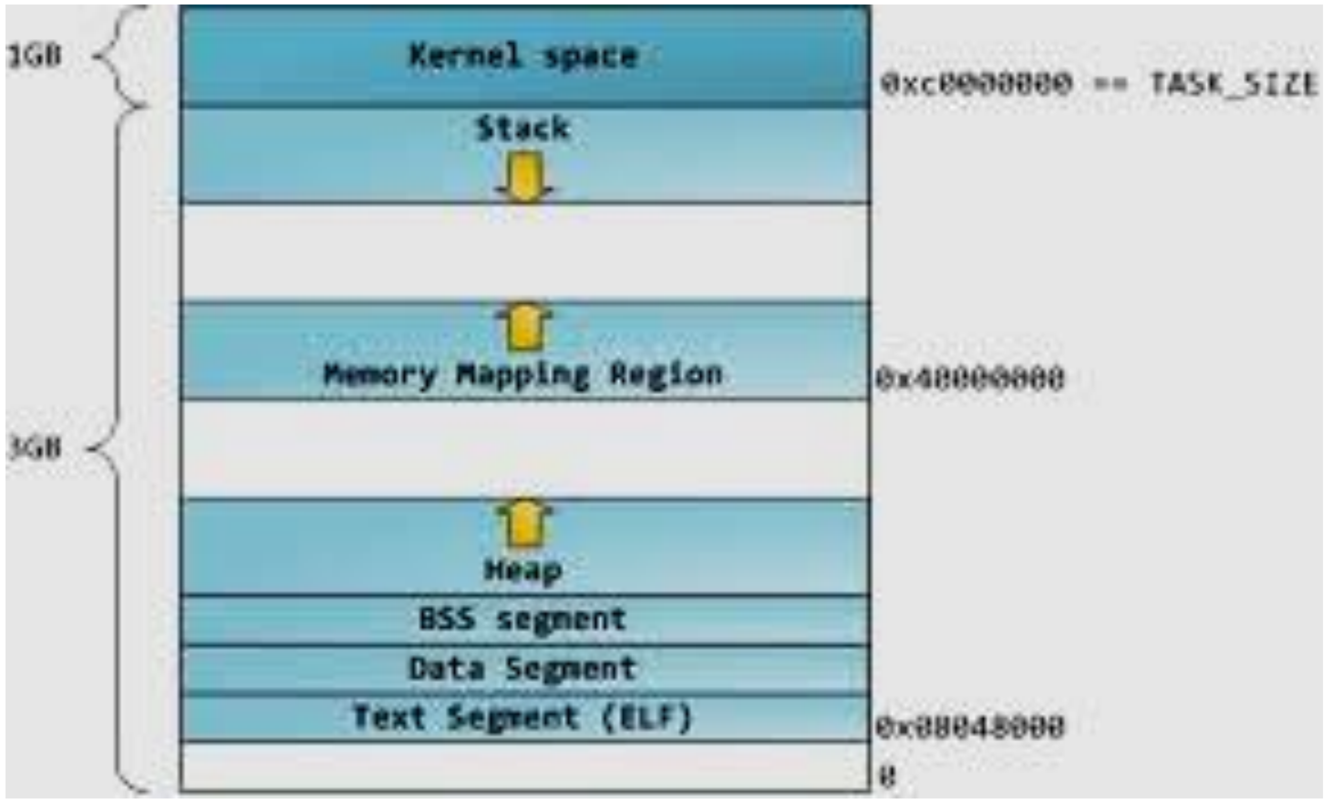
Virtual-8086 Mode While in protected mode, the processor can directly execute real-address mode software such as MS-DOS programs in a safe environment. In other words, if a program crashes or attempts to write data into the system memory area, it will not affect other programs running at the same time. A modern operating system can execute multiple separate virtual-8086 sessions at the same time.

Real-Address Mode Real-address mode implements the programming environment of an early Intel processor with a few extra features, such as the ability to switch into other modes. This mode is useful if a program requires direct access to system memory and hardware devices.

2.2.2 Basic Execution Environment

Address Space

In 32-bit protected mode, a task or program can address a linear address space of up to 4 GBytes. Beginning with the P6 processor, a technique called *extended physical addressing* allows a total of 64 GBytes of physical memory to be addressed. Real-address mode programs, on the other hand, can only address a range of 1 MByte. If the processor is in protected mode and running multiple programs in virtual-8086 mode, each program has its own 1-MByte memory area.



Basic Program Execution Registers

Registers are high-speed storage locations directly inside the CPU, designed to be accessed at much higher speed than conventional memory. When a processing loop is optimized for speed, for example, loop counters are held in registers rather than variables. Figure 2-3 shows the *basic program execution registers*. There are eight general-purpose registers, six segment registers, a processor status flags register (EFLAGS), and an instruction pointer (EIP).

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

EFLAGS Register The EFLAGS (or just *Flags*) register consists of individual binary bits that control the operation of the CPU or reflect the outcome of some CPU operation. Some instructions test and manipulate individual processor flags.

A flag is *set* when it equals 1; it is *clear* (or reset) when it equals 0.

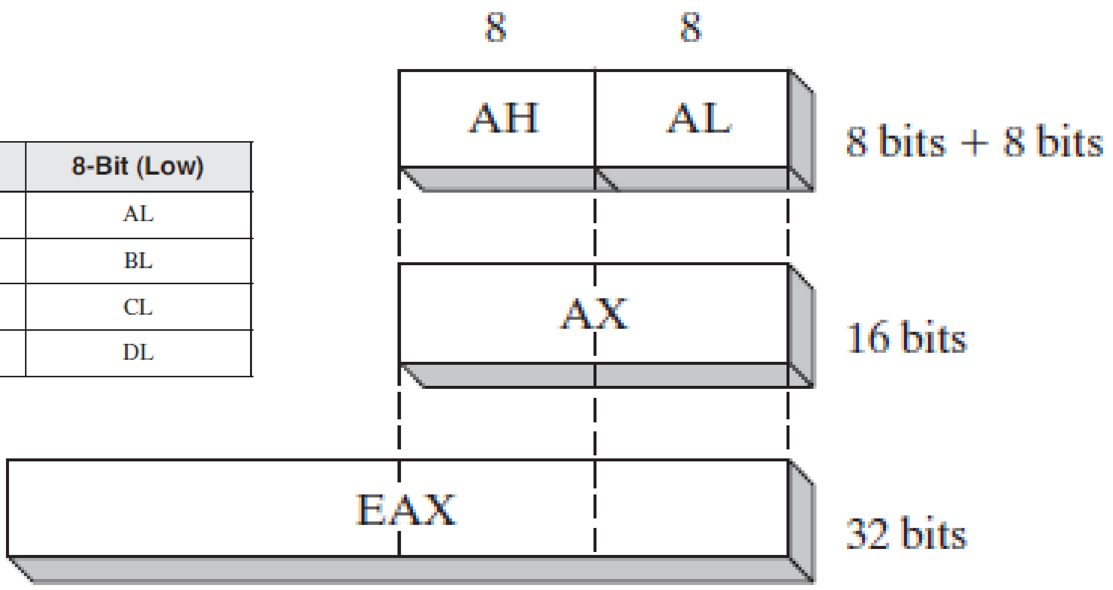
Status Flags The status flags reflect the outcomes of arithmetic and logical operations performed by the CPU. They are the Overflow, Sign, Zero, Auxiliary Carry, Parity, and Carry flags. Their abbreviations are shown immediately after their names:

- The **Carry** flag (CF) is set when the result of an *unsigned* arithmetic operation is too large to fit into the destination.
- The **Overflow** flag (OF) is set when the result of a *signed* arithmetic operation is too large or too small to fit into the destination.
- The **Sign** flag (SF) is set when the result of an arithmetic or logical operation generates a negative result.
- The **Zero** flag (ZF) is set when the result of an arithmetic or logical operation generates a result of zero.
- The **Auxiliary Carry** flag (AC) is set when an arithmetic operation causes a carry from bit 3 to bit 4 in an 8-bit operand.
- The **Parity** flag (PF) is set if the least-significant byte in the result contains an even number of 1 bits. Otherwise, PF is clear. In general, it is used for error checking when there is a possibility that data might be altered or corrupted.

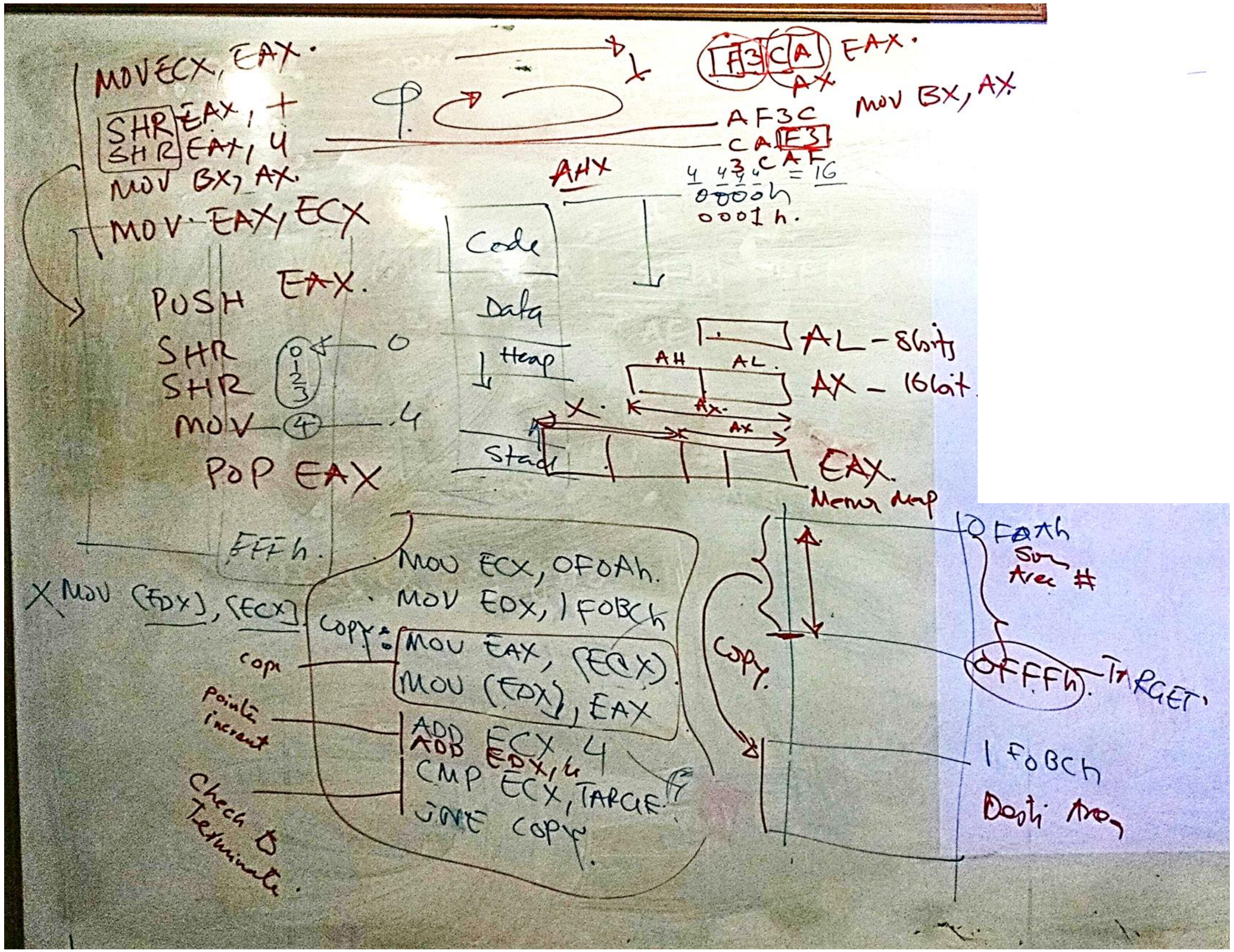
General-Purpose Registers The *general-purpose registers* are primarily used for arithmetic and data movement. As shown in Figure 2-4, the lower 16 bits of the EAX register can be referenced by the name AX.

FIGURE 2–4 General-purpose registers.

32-Bit	16-Bit	8-Bit (High)	8-Bit (Low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL



Portions of some registers can be addressed as 8-bit values. For example, the AX register has an 8-bit upper half named AH and an 8-bit lower half named AL. The same overlapping relationship exists for the EAX, EBX, ECX, and EDX registers:



```
MOV ECX, EAX.
SHR EAX, 4
SHR EAX, 4
MOV BX, AX.
MOV EAX, ECX

PUSH EAX.
SHR
SHR
MOV
POP EAX

X MOV [EDX], [ECX]

MOV ECX, OF0Ah.
MOV EDX, 1FOBCh
copy: MOV EAX, [ECX]
MOV [EDX], EAX
ADD ECX, 4
ADD EDX, 4
CMP ECX, TARGET
JNE copy.
```

- Two memory operands not allowed in x86 ISA.
- Shifting upper 16 bits of 32 bit EAX register so that we can read them from AX register.
- Save value of EAX in ECX register for later retrieval.
- Using PUSH and POP to save and retrieve values of Registers.
- Write a program to copy 4K bytes from memory location starting from 0F0Ah and move them memory location starting with 1FOBCh.

Figure 2–3 Basic program execution registers.

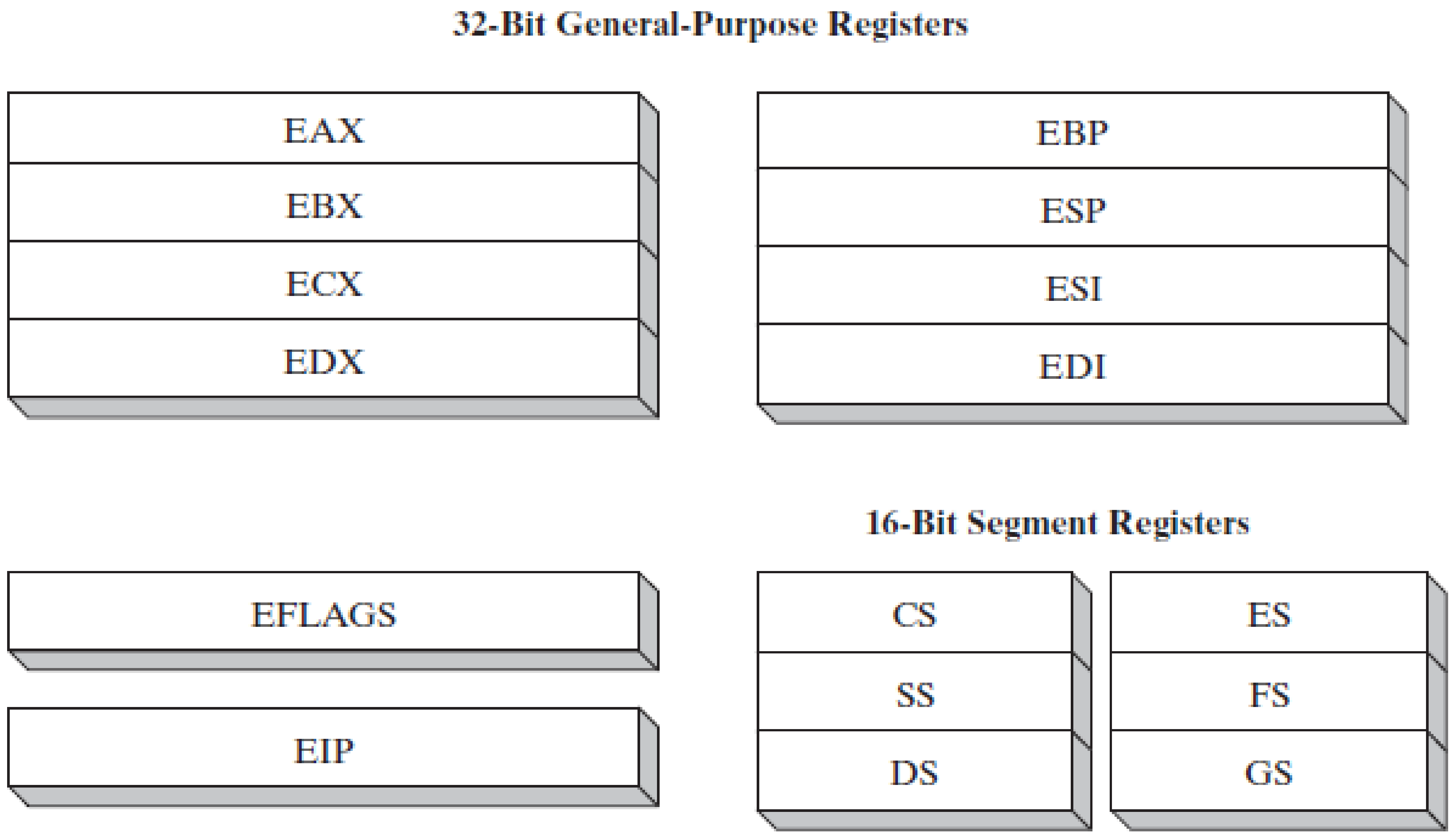


Table 2-1 Operand Sizes in 64-Bit Mode When REX Is Enabled.

Operand Size	Available Registers
8 bits	AL, BL, CL, DL, DIL, SIL, BPL, SPL, R8L, R9L, R10L, R11L, R12L, R13L, R14L, R15L
16 bits	AX, BX, CX, DX, DI, SI, BP, SP, R8W, R9W, R10W, R11W, R12W, R13W, R14W, R15W
32 bits	EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, R8D, R9D, R10D, R11D, R12D, R13D, R14D, R15D
64 bits	RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8, R9, R10, R11, R12, R13, R14, R15

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 is rarely 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.

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