Table 3-1. Implicit Uses of GPRs

Registers[1]				Name	Implicit Uses
Low 8-Bit	16-Bit	32-Bit	64-Bit		
AL	AX	EAX	RAX[2]	Accumulator	• Operand for decimal arithmetic, multiply, divide, string, compare-and-exchange, table-translation, and I/O instructions. • Special accumulator encoding for ADD, XOR, and MOV instructions. • Used with EDX to hold double-precision operands. • CPUID processor-feature information.
BL	BX	EBX	RBX[2]	Base	• Address generation in 16-bit code. • Memory address for XLAT instruction. • CPUID processor-feature information.
CL	CX	ECX	RCX[2]	Count	• Bit index for shift and rotate instructions. • Iteration count for loop and repeated string instructions. • Jump conditional if zero. • CPUID processor-feature information.
DL	DX	EDX	RDX[2]	I/O Address	• Operand for multiply and divide instructions. • Port number for I/O instructions. • Used with EAX to hold double-precision operands. • CPUID processor-feature information.
SIL[2]	SI	ESI	RSI[2]	Source Index	• Memory address of source operand for string instructions. • Memory index for 16-bit addresses.

Note:
1. *Gray-shaded registers have no implicit uses.*
2. *Accessible only in 64-bit mode.*

Table 3-1. Implicit Uses of GPRs (continued)

Registers[1]				Name	Implicit Uses
Low 8-Bit	16-Bit	32-Bit	64-Bit		
DIL[2]	DI	EDI	RDI[2]	Destination Index	• Memory address of destination operand for string instructions. • Memory index for 16-bit addresses.
BPL[2]	BP	EBP	RBP[2]	Base Pointer	• Memory address of stack-frame base pointer.
SPL[2]	SP	ESP	RSP[2]	Stack Pointer	• Memory address of last stack entry (top of stack).
R8B–R10B[2]	R8W–R10W[2]	R8D–R10D[2]	R8–R10[2]	None	No implicit uses
R11B[2]	R11W[2]	R11D[2]	R11[2]	None	• Holds the value of RFLAGS on SYSCALL/SYSRET.
R12B–R15B[2]	R12W–R15W[2]	R12D–R15D[2]	R12–R15[2]	None	No implicit uses

Note:
1. *Gray-shaded registers have no implicit uses.*
2. *Accessible only in 64-bit mode.*

Arithmetic Operations. Several forms of the add, subtract, multiply, and divide instructions use AL or rAX implicitly. The multiply and divide instructions also use the concatenation of rDX:rAX for double-sized results (multiplies) or quotient and remainder (divides).

Sign-Extensions. The instructions that double the size of operands by sign extension (for example, CBW, CWDE, CDQE, CWD, CDQ, CQO) use rAX register implicitly for the operand. The CWD, CDQ, and CQO instructions also uses the rDX register.

Special MOVs. The MOV instruction has several opcodes that implicitly use the AL or rAX register for one operand.

String Operations. Many types of string instructions use the accumulators implicitly. Load string, store string, and scan string instructions use AL or rAX for data and rDI or rSI for the offset of a memory address.

I/O-Address-Space Operations. The I/O and string I/O instructions use rAX to hold data that is received from or sent to a device located in the I/O-address space. DX holds the device I/O-address (the port number).

Table Translations. The table translate instruction (XLATB) uses AL for an memory index and rBX for memory base address.

Compares and Exchanges. Compare and exchange instructions (CMPXCHG) use the AL or rAX register for one operand.

Decimal Arithmetic. The decimal arithmetic instructions (AAA, AAD, AAM, AAS, DAA, DAS) that adjust binary-coded decimal (BCD) operands implicitly use the AL and AH register for their operations.

Shifts and Rotates. Shift and rotate instructions can use the CL register to specify the number of bits an operand is to be shifted or rotated.

Conditional Jumps. Special conditional-jump instructions use the rCX register instead of flags. The JCXZ and JrCXZ instructions check the value of the rCX register and pass control to the target instruction when the value of rCX register reaches 0.

Repeated String Operations. With the exception of I/O string instructions, all string operations use rSI as the source-operand pointer and rDI as the destination-operand pointer. I/O string instructions use rDX to specify the input-port or output-port number. For repeated string operations (those preceded with a repeat-instruction prefix), the rSI and rDI registers are incremented or decremented as the string elements are moved from the source location to the destination. Repeat-string operations also use rCX to hold the string length, and decrement it as data is moved from one location to the other.

Stack Operations. Stack operations make implicit use of the rSP register, and in some cases, the rBP register. The rSP register is used to hold the top-of-stack pointer (or simply, stack pointer). rSP is decremented when items are pushed onto the stack, and incremented when they are popped off the stack. The ENTER and LEAVE instructions use rBP as a stack-frame base pointer. Here, rBP points to the last entry in a data structure that is passed from one block-structured procedure to another.

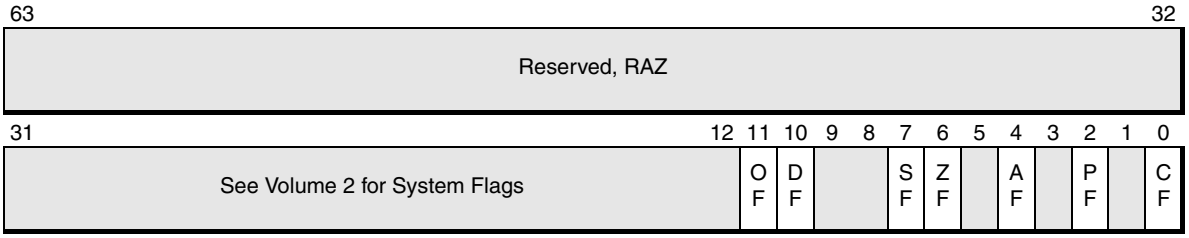
The use of rSP or rBP as a base register in an address calculation implies the use of SS (stack segment) as the default segment. Using any other GPR as a base register without a segment-override prefix implies the use of the DS data segment as the default segment.

The push all and pop all instructions (PUSHA, PUSHAD, POPA, POPAD) implicitly use all of the GPRs.

CPUID Information. The CPUID instruction makes implicit use of the EAX, EBX, ECX, and EDX registers. Software loads a function code into EAX, executes the CPUID instruction, and then reads the associated processor-feature information in EAX, EBX, ECX, and EDX.

3.1.4 Flags Register

Figure 3-5 on page 34 shows the 64-bit RFLAGS register and the flag bits visible to application software. Bits 15–0 are the FLAGS register (accessed in legacy real and virtual-8086 modes), bits 31–0 are the EFLAGS register (accessed in legacy protected mode and compatibility mode), and bits 63–0 are the RFLAGS register (accessed in 64-bit mode). The name *rFLAGS* refers to any of the three register widths, depending on the current software context.



Bits	Mnemonic	Description	R/W
11	OF	Overflow Flag	R/W
10	DF	Direction Flag	R/W
7	SF	Sign Flag	R/W
6	ZF	Zero Flag	R/W
4	AF	Auxiliary Carry Flag	R/W
2	PF	Parity Flag	R/W
0	CF	Carry Flag	R/W

Figure 3-5. rFLAGS Register—Flags Visible to Application Software

The low 16 bits (FLAGS portion) of rFLAGS are accessible by application software and hold the following flags:

- One control flag (the direction flag DF).
- Six status flags (carry flag CF, parity flag PF, auxiliary carry flag AF, zero flag ZF, sign flag SF, and overflow flag OF).

The direction flag (DF) flag controls the direction of string operations. The status flags provide result information from logical and arithmetic operations and control information for conditional move and jump instructions.

Bits 31–16 of the rFLAGS register contain flags that are accessible only to system software. These flags are described in “System Registers” in Volume 2. The highest 32 bits of RFLAGS are reserved. In 64-bit mode, writes to these bits are ignored. They are read as zeros (RAZ). The rFLAGS register is initialized to 02h on reset, so that all of the programmable bits are cleared to zero.

The effects that rFLAGS bit-values have on instructions are summarized in the following places:

- Conditional Moves (CMOV*cc*)—Table 3-4 on page 43.
- Conditional Jumps (J*cc*)—Table 3-5 on page 55.
- Conditional Sets (SET*cc*)—Table 3-6 on page 59.

The effects that instructions have on rFLAGS bit-values are summarized in “Instruction Effects on RFLAGS” in Volume 3.