REP/REPE/REPZ/REPNE/REPNZ—Repeat String Operation Prefix

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
F3 6C	REP INS m8, DX	ZO	Valid	Valid	Input (E)CX bytes from port DX into ES:[(E)DI].
F3 6C	REP INS m8, DX	ZO	Valid	N.E.	Input RCX bytes from port DX into [RDI].
F3 6D	REP INS m16, DX	ZO	Valid	Valid	Input (E)CX words from port DX into ES:[(E)DI.]
F3 6D	REP INS m32, DX	ZO	Valid	Valid	Input (E)CX doublewords from port DX into ES:[(E)DI].
F3 6D	REP INS r/m32, DX	ZO	Valid	N.E.	Input RCX default size from port DX into [RDI].
F3 A4	REP MOVS m8, m8	ZO	Valid	Valid	Move (E)CX bytes from DS:[(E)SI] to ES:[(E)DI].
F3 REX.W A4	REP MOVS m8, m8	ZO	Valid	N.E.	Move RCX bytes from [RSI] to [RDI].
F3 A5	REP MOVS m16, m16	ZO	Valid	Valid	Move (E)CX words from DS:[(E)SI] to ES:[(E)DI].
F3 A5	REP MOVS m32, m32	ZO	Valid	Valid	Move (E)CX doublewords from DS:[(E)SI] to ES:[(E)DI].
F3 REX.W A5	REP MOVS m64, m64	ZO	Valid	N.E.	Move RCX quadwords from [RSI] to [RDI].
F3 6E	REP OUTS DX, r/m8	ZO	Valid	Valid	Output (E)CX bytes from DS:[(E)SI] to port DX.
F3 REX.W 6E	REP OUTS DX, r/m8[1]	ZO	Valid	N.E.	Output RCX bytes from [RSI] to port DX.
F3 6F	REP OUTS DX, r/m16	ZO	Valid	Valid	Output (E)CX words from DS:[(E)SI] to port DX.
F3 6F	REP OUTS DX, r/m32	ZO	Valid	Valid	Output (E)CX doublewords from DS:[(E)SI] to port DX.
F3 REX.W 6F	REP OUTS DX, r/m32	ZO	Valid	N.E.	Output RCX default size from [RSI] to port DX.
F3 AC	REP LODS AL	ZO	Valid	Valid	Load (E)CX bytes from DS:[(E)SI] to AL.
F3 REX.W AC	REP LODS AL	ZO	Valid	N.E.	Load RCX bytes from [RSI] to AL.
F3 AD	REP LODS AX	ZO	Valid	Valid	Load (E)CX words from DS:[(E)SI] to AX.
F3 AD	REP LODS EAX	ZO	Valid	Valid	Load (E)CX doublewords from DS:[(E)SI] to EAX.
F3 REX.W AD	REP LODS RAX	ZO	Valid	N.E.	Load RCX quadwords from [RSI] to RAX.
F3 AA	REP STOS m8	ZO	Valid	Valid	Fill (E)CX bytes at ES:[(E)DI] with AL.
F3 REX.W AA	REP STOS m8	ZO	Valid	N.E.	Fill RCX bytes at [RDI] with AL.
F3 AB	REP STOS m16	ZO	Valid	Valid	Fill (E)CX words at ES:[(E)DI] with AX.
F3 AB	REP STOS m32	ZO	Valid	Valid	Fill (E)CX doublewords at ES:[(E)DI] with EAX.
F3 REX.W AB	REP STOS m64	ZO	Valid	N.E.	Fill RCX quadwords at [RDI] with RAX.
F3 A6	REPE CMPS m8, m8	ZO	Valid	Valid	Find nonmatching bytes in ES:[(E)DI] and DS:[(E)SI].
F3 REX.W A6	REPE CMPS m8, m8	ZO	Valid	N.E.	Find non-matching bytes in [RDI] and [RSI].
F3 A7	REPE CMPS m16, m16	ZO	Valid	Valid	Find nonmatching words in ES:[(E)DI] and DS:[(E)SI].
F3 A7	REPE CMPS m32, m32	ZO	Valid	Valid	Find nonmatching doublewords in ES:[(E)DI] and DS:[(E)SI].
F3 REX.W A7	REPE CMPS m64, m64	ZO	Valid	N.E.	Find non-matching quadwords in [RDI] and [RSI].
F3 AE	REPE SCAS m8	ZO	Valid	Valid	Find non-AL byte starting at ES:[(E)DI].
F3 REX.W AE	REPE SCAS m8	ZO	Valid	N.E.	Find non-AL byte starting at [RDI].
F3 AF	REPE SCAS m16	ZO	Valid	Valid	Find non-AX word starting at ES:[(E)DI].
F3 AF	REPE SCAS m32	ZO	Valid	Valid	Find non-EAX doubleword starting at ES:[(E)DI].

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
F3 REX.W AF	REPE SCAS m64	ZO	Valid	N.E.	Find non-RAX quadword starting at [RDI].
F2 A6	REPNE CMPS m8, m8	ZO	Valid	Valid	Find matching bytes in ES:[(E)DI] and DS:[(E)SI].
F2 REX.W A6	REPNE CMPS m8, m8	ZO	Valid	N.E.	Find matching bytes in [RDI] and [RSI].
F2 A7	REPNE CMPS m16, m16	ZO	Valid	Valid	Find matching words in ES:[(E)DI] and DS:[(E)SI].
F2 A7	REPNE CMPS m32, m32	ZO	Valid	Valid	Find matching doublewords in ES:[(E)DI] and DS:[(E)SI].
F2 REX.W A7	REPNE CMPS m64, m64	ZO	Valid	N.E.	Find matching doublewords in [RDI] and [RSI].
F2 AE	REPNE SCAS m8	ZO	Valid	Valid	Find AL, starting at ES:[(E)DI].
F2 REX.W AE	REPNE SCAS m8	ZO	Valid	N.E.	Find AL, starting at [RDI].
F2 AF	REPNE SCAS m16	ZO	Valid	Valid	Find AX, starting at ES:[(E)DI].
F2 AF	REPNE SCAS m32	ZO	Valid	Valid	Find EAX, starting at ES:[(E)DI].
F2 REX.W AF	REPNE SCAS m64	ZO	Valid	N.E.	Find RAX, starting at [RDI].

NOTES:

1. In 64-bit mode, r/m8 can not be encoded to access the following byte registers if a REX prefix is used: AH, BH, CH, DH.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
ZO	N/A	N/A	N/A	N/A

Description

Repeats a string instruction the number of times specified in the count register or until the indicated condition of the ZF flag is no longer met. The REP (repeat), REPE (repeat while equal), REPNE (repeat while not equal), REPZ (repeat while zero), and REPNZ (repeat while not zero) mnemonics are prefixes that can be added to one of the string instructions. The REP prefix can be added to the INS, OUTS, MOVS, LODS, and STOS instructions, and the REPE, REPNE, REPZ, and REPNZ prefixes can be added to the CMPS and SCAS instructions. (The REPZ and REPNZ prefixes are synonymous forms of the REPE and REPNE prefixes, respectively.) The F3H prefix is defined for the following instructions and undefined for the rest:

- F3H as REP/REPE/REPZ for string and input/output instruction.
- F3H is a mandatory prefix for POPCNT, LZCNT, and ADOX.

The REP prefixes apply only to one string instruction at a time. To repeat a block of instructions, use the LOOP instruction or another looping construct. All of these repeat prefixes cause the associated instruction to be repeated until the count in register is decremented to 0. See Table 4-17.

Table 4-17. Repeat Prefixes

Repeat Prefix	Termination Condition 1*	Termination Condition 2
REP	RCX or (E)CX = 0	None
REPE/REPZ	RCX or (E)CX = 0	ZF = 0
REPNE/REPNZ	RCX or (E)CX = 0	ZF = 1

NOTES:

* Count register is CX, ECX or RCX by default, depending on attributes of the operating modes.

The REPE, REPNE, REPZ, and REPNZ prefixes also check the state of the ZF flag after each iteration and terminate the repeat loop if the ZF flag is not in the specified state. When both termination conditions are tested, the cause of a repeat termination can be determined either by testing the count register with a JECXZ instruction or by testing the ZF flag (with a JZ, JNZ, or JNE instruction).

When the REPE/REPZ and REPNE/REPNZ prefixes are used, the ZF flag does not require initialization because both the CMPS and SCAS instructions affect the ZF flag according to the results of the comparisons they make.

A repeating string operation can be suspended by an exception or interrupt. When this happens, the state of the registers is preserved to allow the string operation to be resumed upon a return from the exception or interrupt handler. The source and destination registers point to the next string elements to be operated on, the EIP register points to the string instruction, and the ECX register has the value it held following the last successful iteration of the instruction. This mechanism allows long string operations to proceed without affecting the interrupt response time of the system.

When a fault occurs during the execution of a CMPS or SCAS instruction that is prefixed with REPE or REPNE, the EFLAGS value is restored to the state prior to the execution of the instruction. Since the SCAS and CMPS instructions do not use EFLAGS as an input, the processor can resume the instruction after the page fault handler.

Use the REP INS and REP OUTS instructions with caution. Not all I/O ports can handle the rate at which these instructions execute. Note that a REP STOS instruction is the fastest way to initialize a large block of memory.

In 64-bit mode, the operand size of the count register is associated with the address size attribute. Thus the default count register is RCX; REX.W has no effect on the address size and the count register. In 64-bit mode, if 67H is used to override address size attribute, the count register is ECX and any implicit source/destination operand will use the corresponding 32-bit index register. See the summary chart at the beginning of this section for encoding data and limits.

REP INS may read from the I/O port without writing to the memory location if an exception or VM exit occurs due to the write (e.g., #PF). If this would be problematic, for example because the I/O port read has side-effects, software should ensure the write to the memory location does not cause an exception or VM exit.

Operation

```
IF AddressSize = 16
   THEN
      Use CX for CountReg;
      Implicit Source/Dest operand for memory use of SI/DI;
   ELSE IF AddressSize = 64
      THEN Use RCX for CountReg;
      Implicit Source/Dest operand for memory use of RSI/RDI;
   ELSE
      Use ECX for CountReg;
      Implicit Source/Dest operand for memory use of ESI/EDI;
FI;
WHILE CountReg ≠ 0
    DO
         Service pending interrupts (if any);
         Execute associated string instruction;
         CountReg := (CountReg – 1);
         IF CountReg = 0
               THEN exit WHILE loop; FI;
         IF (Repeat prefix is REPZ or REPE) and (ZF = 0)
         or (Repeat prefix is REPNZ or REPNE) and (ZF = 1)
               THEN exit WHILE loop; FI;
    OD;
```

Flags Affected

None; however, the CMPS and SCAS instructions do set the status flags in the EFLAGS register.

Exceptions (All Operating Modes)

Exceptions may be generated by an instruction associated with the prefix.

64-Bit Mode Exceptions

#GP(0)	If the memory address is in a non-canonical form.