

## Transfer				
이름	설명	코드	작동	플래그
---  ---  ---  ---  ---				
MOV	이동 (복사)	MOV 목적지, 소스	목적지 := 소스	O D I T S Z A P C
XCHG	교환	XCHG Op1, Op2	Op1 := Op2, Op2 := Op1	
STC	Carry 설정	STC	CF := 1	1
CLC	Carry 초기화	CLC	CF := 0	0
CMC	Carry 보완	CMC	CF := ¬CF	±
STD	방향 설정	STD	DF := 1 (문자열 연산 하향)	1
CLD	방향 초기화	CLD	DF := 0 (문자열 연산 상향)	0
STI	중단 설정	STI	IF := 1	1
CLI	중단 초기화	CLI	IF := 0	0
## Stack Manipulation				
이름	설명	코드	작동	플래그
---  ---  ---  ---  ---				
PUSH	스택에 푸시	PUSH 소스	[SP] := 소스	
POP	스택에서 꺼내기	POP 목적지	목적지 := [SP], SP := SP + 1	
PUSHF	스택에 푸시	PUSHF	[SP] := [SP] + 1, SP := SP	
POPF	스택에서 꺼내기	POPF	[SP] := [SP] + 1, SP := SP	
CWD	16비트 레지스터 팝	CWD	AX := AX	
CWDE	32비트 레지스터 팝	CWDE	EAX := AX	
IN	입력	IN 목적지, 포트	AL/AX/EAX := 지정된 포트의 byte/word/double	
OUT	출력	OUT 포트, 소스	지정된 포트의 byte/word/double := AL/AX/EAX	
## Arithmetic Operations				
이름	설명	코드	작동	플래그
---  ---  ---  ---  ---				
ADD	덧셈	ADD 목적지, 소스	목적지 := 목적지 + 소스	± ± ± ±
ADG	Carry와 함께 업셋	ADG 목적지, 소스	목적지 := 목적지 + 소수 + CF	± ± ± ± ± ±
SUB	뺄셈	SUB 목적지, 소수	목적지 := 목적지 - 소수	± ± ± ± ±
SBB	빌려서 뺄셈	SBB 목적지, 소수	목적지 := 목적지 - (소수 + CF)	± ± ± ± ±
DIV	나눗셈 (부호 있음)	DIV Op	Op = byte: AL := AX / Op	
DIV 386	나눗셈 (부호 없음)	DIV Op	Op = double: EAX := EDX:EAX / Op	?
IDIV	부호 있는 정수 나눗셈	IDIV Op	Op = byte: AL := AX / Op	?
IDIV	부호 있는 정수 나눗셈	IDIV Op	Op = word: AX := DX:AX / Op	?
IDIV 386	부호 있는 정수 나눗셈	IDIV Op	Op = double: EAX := EDX:EAX / Op	?
MUL	곱셈 (부호 없음)	MUL Op	Op = byte: AX := AL * Op	?
MUL	곱셈 (부호 없음)	MUL Op	Op = word: DX:AX := AX * Op	?
MUL 386	곱셈 (부호 없음)	MUL Op	Op = double: EDX:EAX := EAX * Op	?
IMUL	부호 있는 정수 곱셈	IMUL Op	Op = byte: AX := AL * Op	?
IMUL	부호 있는 정수 곱셈	IMUL Op	Op = word: DX:AX := AX * Op	?
IMUL 386	부호 있는 정수 곱셈	IMUL Op	Op = double: EDX:EAX := EAX * Op	?
INC	증가	INC Op	Op := Op + 1 (Carry는 영향을 받지 않음)	±
DEC	감소	DEC Op	Op := Op - 1 (Carry는 영향을 받지 않음)	±
CMP	비교	CMP Op1, Op2	Op1 - Op2	± ± ± ±
SAL	산술 왼쪽 시프트 (= shl)	SAL Op, 양수	Op := Op << 양수	± ± ± ±
SAR	산술 오른쪽 시프트	SAR Op, 양수	Op := Op >> 양수	± ± ± ±
RCL	Carry를 통한 왼쪽 회전	RCL Op, 양수	Op := Op << 양수	± ± ± ±
RCR	Carry를 통한 오른쪽 회전	RCR Op, 양수	Op := Op >> 양수	± ± ± ±
ROL	왼쪽 회전	ROL Op, 양수	Op := Op << 양수	± ± ± ±
ROR	오른쪽 회전	ROR Op, 양수	Op := Op >> 양수	± ± ± ±
## Logic Operations				
이름	설명	코드	작동	플래그
---  ---  ---  ---  ---				
NEG	부정 (2의 보수)	NEG Op	Op := ¬Op	± ± ± ±
NOT	반전 (각 비트)	NOT Op	Op := ¬(Op (비트 변환))	
AND	논리 AND	AND 목적지, 소수	목적지 := 목적지 ∧ 소수	0
OR	논리 OR	OR 목적지, 소수	목적지 := 목적지 ∨ 소수	0
XOR	논리 XOR	XOR 목적지, 소수	목적지 := (exor) 소수	0
SHL	논리 왼쪽 시프트 (= SAL)	SHL Op, 양수	Op := Op << 양수	±
SHR	논리 오른쪽 시프트	SHR Op, 양수	Op := Op >> 양수	±

자세한 정보는 명령어 사양을 참조하십시오

- 그러면 CF := 0, OF := 0, 그렇지 않으면 CF := 1, OF := 1

이름	설명	코드	작동	플래그
NEG	부정 (2의 보수)	NEG Op	Op := ¬Op	± ± ± ±
NOT	반전 (각 비트)	NOT Op	Op := ¬(Op (비트 변환))	
AND	논리 AND	AND 목적지, 소수	목적지 := 목적지 ∧ 소수	0
OR	논리 OR	OR 목적지, 소수	목적지 := 목적지 ∨ 소수	0
XOR	논리 XOR	XOR 목적지, 소수	목적지 := (exor) 소수	0
SHL	논리 왼쪽 시프트 (= SAL)	SHL Op, 양수	Op := Op << 양수	±
SHR	논리 오른쪽 시프트	SHR Op, 양수	Op := Op >> 양수	±

자세한 정보는 명령어 사양을 참조하십시오

- 그러면 CF := 0, OF := 0, 그렇지 않으면 CF := 1, OF := 1