<어셈블리 명령어 정리>

INC(increase)
: 피연산자에 1을 더함
ex) INC reg

DEC(decrease)
: 피연산자에 1을 뺌
ex) DEC reg

ADD(add)
: destination에 source 값을 더하여 destination에 저장
ex) ADD destination, source

SUB(subtract)
: destination에 source 값을 빼서 destination에 저장
ex) SUB destination, source

MUL(unsigned integer multiply)
: 부호 없는 al, ax, eax의 값을 피연산자와 곱함
  피연산자가 8비트 - al과 곱해 ax에 저장, 16비트 - ax와 곱해 dx:ax에 저장
ex) MUL reg

IMUL(integer multiply)
: 부호 있는 al, ax, eax의 값을 피연산자와 곱함
ex) IMUL destination, value // value를 al, ax, eax와 곱해 destination에 저장
    IMUL destination, value, value // value끼리 곱해 destination에 저장

DIV(unsigned integer divide)
: 8, 16, 32 비트의 부호 없는 정수 나눗셈
ex) DIV reg

MOV(move)
: source에서 destination으로 데이터를 복사
ex) MOV destination, source

MOVS(move string)
: source에서 destination으로 데이터 복사
ex) MOVS destination, source

MOVSB, MOVSW, MOVSD(move string)
: SI또는 ESI 레지스터에 의해 지정된 메모리 주소의 내용을 DI 또는 EDI 레지스터에 의해 지정되는 메모리 주소로 복사
B - byte 단위 복사, W - word 단위 복사 D - dword 단위 복사
DF(방향 플래그)가 1 - ESI와 EDI는 복사 시에 감소
DF가 0 - ESI와 EDI는 복사 시에 증가
ex) MOVSB
MOVSW
MOVSD

INT(interrupt)
: 소프트웨어 인터럽트를 발생시켜 운영체제의 서브루틴을 호출
ex) INT imm

AND(logical AND)
: destination과 source 피연산자의 각 비트 and 연산
ex) AND reg, reg

OR(inclusive OR)
: destination과 source 피연산자의 각 비트 or 연산
ex) OR reg, reg

XOR(exclusive OR)
: destination과 source 피연산자의 각 비트 xor 연산
ex) XOR reg, reg // 레지스터를 0으로 초기화 시킬 때 많이 사용

TEST(test)
: 두 피연산자 사이에 논리적인 and 연산을 수행하여 플래그 레지스터에 영향을 주지만, 결과값은 저장하지 않음
ex) TEST reg, reg

PUSH(push on stack)
: 스택에 값을 넣음. esp의 값이 4만큼 줄어들고, 이 위치에 새로운 값이 채워짐
ex) PUSH reg16

PUSHAD(push all)
: eax, ebx, ecx, edx, esi, edi, esp, ebp 레지스터의 값을 스택에 push
레지스터들의 값을 보관해야 할 때 사용
ex) PUSHAD

PUSHFD(push flags)
: 플래그 레지스터를 스택에 push함
  플래그 레지스터의 값을 보관해야 할 필요가 있을 때 사용
ex) PUSHFD

POP(pop from stack)
: esp 레지스터가 가리키고 있는 위치의 스택 공간에서 4byte 만큼을 destination 피연산자에 복사하고, esp 레지스터의 값에 4를 더함
ex) POP destination

POPAD(pop all flags from stack)
: 스택에 존재하는 값을 eax, ebx, edx, esi, edi, esp, ebp 레지스터로 pop함
  PUSHAD 명령어로 스택에 보관해 놓은 레지스터 정보를 다시 이용하려고 할 때 사용
ex) POP destination

POPFD(pop flags from stack)
: 스택에 존재하는 값을 플러그 레지스터로 pop함
  PUSHFD 명령어로 스택에 보관해놓은 레지스터 정보를 다시 이용하려고 할 때 사용
ex) POPFD

XCHG(exchange)
: 두 피연산자의 내용이 서로 교환
ex) XCHG reg, reg

NEG(negate)
: 피연산자의 2의 보수를 계산하여 결과를 피연산자에 저장
ex) NEG reg

PTR
: 피연산자의 크기를 재설정
ex) MOV eax, DWORD PTR value // value의 크기를 DWORD 크기로 재설정하여 eax에 복사

OFFSET
: 세그먼트의 시작으로부터 변수가 위치한 거리까지의 상대적 거리를 반환
ex) MOV esi, OFFSET value // value가 존재하는 위치를 세그먼트 시작 지점부터의 상대적 거리로 구해서 esi 레지스터에 복사

LEA(load effective address)
: source 피연산자의 유효 주소를 계산하여 destination 피연산자에 복사 (주소를 알아내서 복사하는 명령어)
ex) LEA reg, mem

REP(repeat string)
: ecx 레지스터를 카운터로 사용해서 문자열 관련 명령을 ecx>0인 동안 반복
ex) REP MOVS destination, source

JMP(jump unconditionally to lable)
: 피연산자의 위치로 실행 흐름이 변경. 피연산자가 가리키는 코드로 넘어가 실행
ex) JMP reg16

CALL(call a procedure)
: 함수 호출시 사용. JMP 명령어와 같이 프로그램의 실행 흐름이 변경되지만, JMP와 달리 돌아올 리턴 어드레스를 스택에 저장함.
ex) CALL 함수 주소

CMP(compare)
: 두 피연산자를 비교하는 작업. destination 피연산자에서 source 피연산자를 묵시적으로 빼서 값을 비교. 같으면 ZF(zero flag)가 1로 설정, 다르면 0으로 설정됨.
ex) CMP reg, reg

NOP(no operation)
: 아무 일도 하지 않는 명령어.
ex) NOP

<레지스터 정리>

EAX
: 누산기, 곱셈과 나눗셈 연산에서 자동으로 사용

EBX
: 베이스 레지스터, 특정 주소를 지정

ECX
: 루프 카운터

EDX
: 데이터 레지스터, 입출력 연산에서 반드시 간접 주소 지정에 사용

EBP
: 베이스 포인터, 스택의 데이터에 접근하기 위해 사용

ESP
: 스택 포인터, 현재까지 사용된 스택의 위치 저장. 스택 최상부의 오프셋 가리킴

ESI
: 읽기 인덱스, 문자열 전송이나 비교에서 사용되는데, 주로 소스 문자열의 오프셋 가리킴

EDI
: 쓰기 인덱스

EIP
: 명령어 포인터 레지스터, 실행할 다음 명령어의 주소 포함

EFLAGS
: CPU의 동작을 제어하거나 CPU 연산의 결과를 반영

https://hongci.tistory.com/19

https://beomnaegol.tistory.com/entry/Assembly-%EB%A0%88%EC%A7%80%EC%8A%A4%ED%84%B0%EC%9D%98-%EC%A2%85%EB%A5%98%EC%99%80-%ED%8A%B9%EC%A7%95