2주차 3차시. 어셈블리어 기본 명령,스택을 통한 명령처리 과정

【학습목표】

1. 스택 동작 이해, 셸 코드 생성에 대해 설명할 수 있다.

학습내용1 : 어셈블리어의 기본 명령

1. 산술 연산 명령

* 기본적인 정수 계산
* ADD(Add)
제1피연산자와 제2피연산자 값을더한 결과 값을 제1피 연산자에 저장
ADD AL 4
* SUB(Subtract)
제1피연산자에서 제2피연산자 값을뺀 결과 값을 제1피연산자에 저장
SUB AL 4
* CMP(Compare)
데이터의 두 값 비교 시 사용
CMP A, B 의 경우 A에서 B를 뺀 값이 0이면 참
CMD AL 4

2. 기타 산술 연산 명령

명령		설명
ADC	Add with Carry	캐리를 포함한 덧셈 수행
SBB	Subtraction with borrow	캐리를 포함한 뺄셈 수행
DEC	Decrement	피연산자 내용을 하나 감소 시킴
NEG	Change Sign	피연산자의 부호 반전(2의 보수)
INC	Increment	피연산자 내용을 하나 증가 시킴

명령		설명
AAA	ASCII adjust for add	덧셈 결과의 AL 값을 UNPACK 10진수로 보정
DAA	Decimal adjust for add	덧셈 결과의 AL 값을 PACK 10진수로 보정
AAS	ASCII adjust for subtract	뺄셈 결과의 AL값을 UNPACK 10진수로 보정

명령		설명
DAS	Decimal adjust for subtract	뺄셈 결과의 AL값을 PACK 10진수로 보정
MUL	Multiply(Unsigned)	AX와 피연산자의 곱셈 결과를 AX 또는 DX:AX에 저장
IMUL	Integer Multiply(Signed)	부호화된 곱셈을 수행
AAM	ASCII adjust for Multiply	곱셈 결과의 AX 값을 UNPACK 10진수로 보정

명령		설명
DIV	Divide(Unsigned)	AX 또는 DX:AX 내용을 피연산자로 나눔 몫은 AL 또는 AX에 저장하고, 나머지는 AH 또는 DX에 저장
IDIV	Integer Divide(Signed)	부호화된 나눗셈
AAD	ASCII adjust for Divide	나눗셈 결과 AX 값을 UNPACK 10진수로 보정
CBW	Convert byte to word	AL의 바이트 데이터를 부호 비트를 포함하여 AX 워드로 확장

3. 데이터 전송 명령

* 메모리, 범용 레지스터, 세그먼트 레지스터로참조되는 주소에 존재하는 데이터 전송
* MOV(Move)

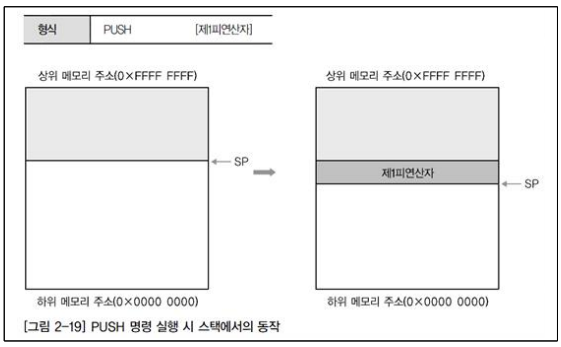
데이터 이동할 때 사용

MOV AX [BP+8]

* PUSH(Push)

스택에 데이터를 삽입할 때 사용

[그림 2-19]와 같이 스택은 커지고,스택 포인터는 데이터 크기만큼 감소



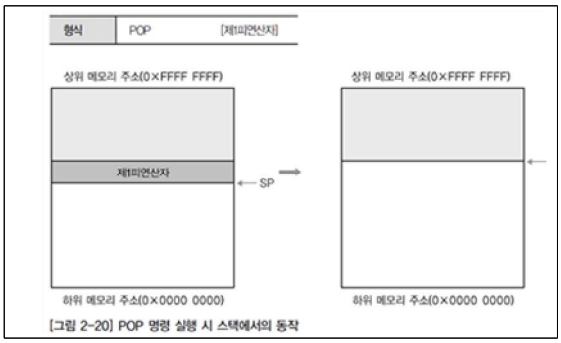
[그림 2-19] PUSH 명령 실행 시 스택에서의 동작

* POP(Pop)

스택에서 데이터 삭제할 때 사용

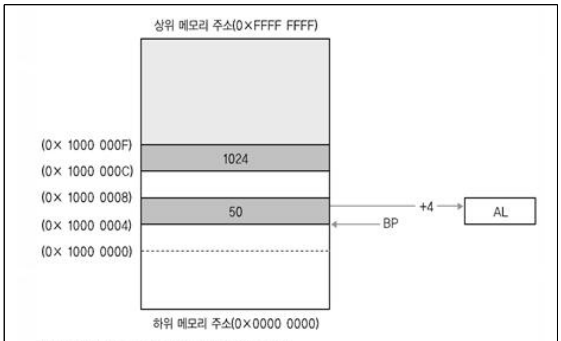
스택에서 삭제된 명령은 반환 값으로 받아 사용 가능

[그림2-20]과 같이 스택 포인터는삭제하는 데이터 크기만큼증가



[그림 2-20] POP 명령 실행 시 스택에서의 동작

* LEA(Load effective address to register)
데이터의 값 이동할 때 사용
제1피연산자는 데이터 이동의 목적지
제2피연산자에'BP+4'가 있을 경우'BP'만 주소 값으로 인식 '+4'는BP 주소 값에 대한추가 연산으로 처리



[그림 2-21] LEA 명령 실행 시 스택에서의 동작

4. 기타 데이터 전송 명령

명령		설명
XCHG	Exchange Register/ Memory with Register	첫 번째 피연산자와 두 번째 피연산자를 교환
IN	Input from AL/ AX to Fixed port	피연산자로 지시된 포트로 AX에 데이터를 입력

명령		설명
OUT	Output from AL/AX to Fixed port	피연산자가 지시한 포트로 AX의 데이터를 출력
XLAT	Translate byte to AL	BX:AL이 지시한 테이블의 내용을 AL로 로드
LDS	Load Pointer to DS	LEA 명령과 유사한 방식으로 다른 DS 데이터의 주소의 내용을 참조 시 사용

명령		설명
LES	Load Pointer to ES	LEA 명령과 유사한 방식으로 다른 ES 데이터의 주소의 내용을 참조 시 사용
LAHF	Load AH with Flags	플래그의 내용을 AH의 특정 비트로 로드
SAHF	Store AH into Flags	AH의 특정 비트를 플래그 레지스터로 전송

명령		설명
PUSHF	Push Flags	플래그 레지스터의 내용을 스택에 삽입
POPF	Pop Flags	스택에 저장되어 있던 플래그 레지스터 값을 삭제

5. 논리 명령

* 연산부호가 논리연산을 지정하는 명령
* AND(And)
대응되는 비트가 둘 다 1일 때만 결과가 1
그 이외는 모두 0
AND AX 10h

AX	1	0	0	0
0×10h	1	0	1	0
AND 연산 결과	1	0	0	0

* OR(Or)
대응되는 비트 중 하나만 1이어도 결과가 1
둘 다 0인 경우에만 0
OR AX 10h

AX	1	0	0	0
0×10h	1	0	1	0
OR 연산 결과	1	0	1	0

* XOR(Exclusive Or)
대응되는 비트 중에서 한 비트가1이고 다른 비트가 0이면
1 두 개의 비트가 모두 0 또는 1일 때 0
XOR AX 10h

AX	1	0	0	0
0×10h	1	0	1	0
XOR 연산 결과	0	0	1	0

* NOT(Invert)
피연산자의 1의 보수를 구하는 작동 코드
각 비트를 반전
NOT AX

AX	1	0	0	0
NOT 연산 결과	0	1	1	1

* TEST(And Function to flags, no result)
데이터의 두 값 비교할 때 사용
데이터의 변경 없이 단순 비교
TEST AL 00001001b

AL	1	0	0	0
00001001b	1	0	0	1

기타 논리 명령

명령		설명
SHL / SAL	Shift Left/arithmetic Left	왼쪽으로 피연산자만큼 자리 이동
SHR /SAR	Shift Right / Shift arithmetic Right	오른쪽으로 피연산자만큼 자리 이동
ROL	Rotate Left	왼쪽으로 피연산자만큼 회전 이동

명령		설명
ROR	Rotate Right	오른쪽으로 피연산자만큼 회전 이동
RCL	Rotate through carry left	자리올림(Carry)을 포함하여 왼쪽으로 피연산자만큼 회전 이동
RCR	Rotate through carry Right	자리올림(Carry)을 포함하여 오른쪽으로 연산자 만큼 회전이동

6. 스트링 명령

* 바이트로 구성된 스트링(strings of bytes)을메모리 내에서 전송
* REP(Repeat)
ADD나 MOVS와 같은 작동 코드의 앞에 위치
CX가 0이 될 때까지 뒤에 오는 스트링 명령 반복
REP 작동 코드
* MOVS(Move String)
바이트나 워드, 더블워드 옮기는 명령
MOVSB, MOVSW, MOVSD 명령 존재
DS:SI가 지시한 메모리 데이터를ES :DI가 지시한 메모리로 전송

* MOVS(Move String) 사용의 예

형식	MOVSB
사용 예	CLD LEA SI, String_1 LEA DI, String_2 MOV CX, 384 REP MOVSB - CLD (Clear Direction) : 디렉션 플래그를 지움 - LEA SI, String_1 : String 1의 주소 값을 SI(Source Index)에 저장 - LEA DI, String_2 : String 2의 주소 값을 DI(Destination Index)에 저장 - MOV CX, 384 : CX에 384 저장 - REP MOVSB : SI로부터 DI까지 CX가 0이 될 때까지 1바이트씩 복사

* 기타 스트링 명령

명령문		설명
CMPS	Compare String	DS: SI와 ES: DI의 내용을 비교한 결과에 따라 플래그를 설정
SCAS	Scan String	AL 또는 AX와 ES:DI가 지시한 메모리 내용을 비교한 결과에 따라 플래그를 설정
LODS	Load String	SI 내용을 AL 또는 AX로 로드
STOS	Store String	AL 또는 AX를 ES: DI가 지시하는 메모리에 저장

7. 제어 전송 명령

* 프로그램의 흐름 제어

* JMP(Unconditional Jump)

대표적인 점프 명령 프로그램을실행할 주소 또는 라벨로 이동

JMP 100h

* 조건부 점프 명령

명령		점프 조건	설명
JC	carry flag set	CF = 1 CF	값이 1이면 점프
JNC	not carry flag set	CF = 0 CF	값이 0 이면 점프
JE/JZ	Equal / Zero	ZF = 1	결과가 0이면 점프
JA/JNBE	above/not below nor equal	CF = 0 and ZF = 0	결과가 크면 점프 (부호화 안된 수)

명령		점프 조건	설명
JAE/JNB	above or equal/not below	CF = 0	결과가 크거나 같으면 점프(부호화 안된 수)
JB/JNAE	below/not above nor equal	CF = 1	결과가 작으면 점프 (부호화 안된 수)
JL/JNGE	less/not greater nor equal	SF ! = OF	결과가 작으면 점프 (부호화된 수)

명령		점프 조건	설명
JBE/JNA	below or equal/ not above	(CF or ZF) = 1	결과가 작거나 같으면 점프(부호화 안된 수)
JG/JNLE	greater/ not less nor equal	ZF = 0 and SF = OF	결과가 크면 점프 (부호화 안된 수)
JGE/JNL	greater or equal/not less	SF = OF	결과가 크거나 같으면 점프(부호화된 수)

명령		점프 조건	설명
JLE/JNG	less or equal/not greater	ZF = 1 or SF ! = OF	결과가 작거나 같으면 점프(부호화 안된 수) 결과가 0이 아니면 점프
JNE/JNZ	not equal/not zero	ZF = 0	JNOnot overflowOF=0오버플로우가 아닌 경우 점프

명령		점프 조건	설명
JNS	not sign	SF = 0	SF가 1이면 점프
JO	overflow	OF = 1	오버플로우 발생 시 점프
JP/JPE	parity/ parity even	PF = 1	PF가 1이면 점프
JS	Sign	SF = 1	SF가 1이면 점프
JCXZ	CX Zero	CX = 0	CX가 0이면 점프

* CALL(Call)
JMP처럼 함수 호출할 때 사용
제1피연산자에 라벨을 지정
리턴 주소로 IP(Instruction Pointer) 주소 백업
PUSH EIP + JMP 와 같은 의미
CALL [제1피연산자]

* RET(Return from CALL)
함수에서 호출한 곳으로 돌아갈 때 사용하는 명령
POP EIP 와 같은 의미
CALL [제1피연산자]

LOOP(Loop CX times)
문장들의 블록을 지정된 횟수만큼 반복
CX는 자동적으로 카운터로 사용되며루프를 반복할 때 마다 감소

형식	LOOP [제1피연산자]

사용 예	MOV AX, 0
	MOV CX, 5
	L1 : INC AX
	LOOP L1

	제1피연산자는 라벨이 된다. 예에서 L1이 CX의 숫자만큼 5번 회전하므로, 결과적으로 AX는 5가 된다.

* INT(Interrupt)
인터럽트가 호출되면CS:IP(Code Segment : Instruction Pointer)와플래그를 스택에 저장
그 인터럽트에 관련된 서브 루틴이 실행
CALL [제1피연산자]

* 기타 제어 전송 명령

명령		설명
INTO	Interrupt on Overflow	오버플로우 발생 시 인터럽트 실행
IRET	Interrupt Return	인터럽트에서 복귀(리턴)
LOOPZ/LOOPE	Loop while Zero/Equal	제로 플래그가 1이고 CX 값이 0보다 크면 루프를 계속 수행
LOOPNZ/LOOPNE	Loop while not Zero/not Equal	제로 플래그가 0이고 CX 값이 0보다 크면 루프를 계속 수행

8. 프로세스 제어 명령

*레지스터의 플래그 값을 세트 또는 클리어하는 명령
*STC(Set Carry)
피연산자 없이 사용
EFLAGS 레지스터의 CF 값을 세트
*NOP(No Operation)
아무 의미 없는 명령
일종의 빈 칸을 채우려고 사용

* 기타 프로세스 제어 명령

명령		설명
CLC	Clear Carry	캐리 플래그를 클리어
CMC	Complement Carry	캐리 플래그를 반전
HLT	Halt	정지
CLD	Clear Direction	디렉션 플래그를 클리어
CLI	Clear Interrupt	인터럽트 플래그를 클리어

명령		설명
STD	Set Direction	디렉션 플래그를 세트
STI	Set Interrupt	인터럽트 인에이블 플래그를 세트
WAIT	Wait	프로세스를 일시 정지 상태로 만듬
ESC	Escape to External device	종료 명령

【학습정리】

1. 어셈블리어 기본 명령은 산술 연산 명령, 데이터 전송 명령, 논리 명령, 스트링 명령, 제어 전송 명령, 프로세서 제어 명령이 있다.