아직도 어셈블리언어가 필요한가.

어셈블리 언어의 지식은 고속연산 루틴을 작성하기위해서만 필요한 것이 아니라.마이크로 컴퓨터의 시스템을 잘알기 위하여 필요한것이다.
메모리상의 데이터나 I/O 기기를 직접 액세스 하는등, 고급언어에서는 할수없는 조작을 할수있다 라는 것이다.
CPU 가 이해할수있는 것은 수치로 기술된 명령 밖에 이해 할수가 없읍니다.

의사 명령이 확충된 매크로어셈블러:
직접기계어로는 번역되지는 않지만 번역 할때 어셈블러에 의해 참조 되는 것입니다.

어셈블리 언어란: 기계어와 1 대 1 로 대응된 명령을 기술하는 언어
어셈블러란: 어셈블리 언어로 쓰여진 소스를 번역 기계어 프로그램을 작성해주는프로그램.
MASM 은 모듈별 개발이 용이하므로 축적된 소프트웨어를 유효하게 활용 할수있다.

MASM 에 의한 어셈블러 개발법 : 아스키 문자열로써 저장하는 형태의 에디터라면 무엇이든 사용 할수있다.

링커의 역할:
몇가지 의 화일을 합쳐서 하나의 프로그램으로 만드는 것이 링커의 역활이고 링커에 입력이 되는 화일이 중간 화일인 오브젝트 화일 입니다.

라이브러리의 사용법:
모듈별로 개발된 오브젝트화일 화일 을 합쳐서 하나의 라이브러리 화일 이라는 것을 작성 해둡니다. 링크시에 라이브러리 를 지정하는 것만으로도 그중에서 필요한 오브젝트 화일만을 자동으로 꺼내어져 결합하는 것이 가능합니다.

CPU는 기계어를 어떻게 이해하나

컴파일러 언어: 기계어로의 번역작업을 한꺼번에 합쳐서 수행해 버린뒤에 실행
인터프리터 언어:조금씩 번역하면서 실행
1 이나 0 의 정보의 양을 1 비트(bit)
8비트를 한조로 1바이트(byte)

16 비트 CPU 란? :
CPU 내의 레지스터가 16비트의 크기,주변장치와 데이터의 교환을 하는 데이터 버스가 16개있다. (16 비트 = 2 바이트 = 1워드)

CPU 의 동작은 2 바이트 를 하나의 단위로 서 실행하고 있어 워드(= 2바이트) 단위로 데이터를 조작 하는 경우의 쪽이 효율이 좋게 되도록 섥계되어 있으므로 효율이 좋은 프로그램을 작성 할때는 이것을 고려 해볼 필요가 있다.
특히 ,데이터를 액세스 할때는 0 번지와 1번지 2번지와 3번지 이러한 식으로 짝수 홀수 순으로 조합하여 실행하면 , 한번에 두바이트의 데이터를 읽기 쓰기가 가능하므로 처리의 효율이 향상됩니다.

8068 의 레지스터의 구성

AX:	AH	AL	Accumulator	Register	General-purpose
BX:	BH	BL	Base	Register	Register
CX:	CH	CL	Counter	Register	
DX:	DH	DL	Data	Register	
		SP	Stack	Pointer	
		BP	Base	Pointer	
		SI	Source	Index	
		DI	Destination	Index (선행 인덱스)	
		IP	Instruction	Pointer	
	FLAGS H	FLAGS L	Flag	Register	
		CS	Code Segment	Register	Segment Register
		DS	Data Segment	Register	
		SS	Stack Segment	Register	
		ES	Extra Segment	Register	

*.주의
CS:코드세그먼트-CPU가 실행해야 될 명령이 저장되어 있는 세그먼트의 시작을 나타냄

SI:소스인덱스 - 번지의 간접지정에 사용 , 특히 스트링명령에 있어서는
메모리로 부터 레지스터로 데이터를 전송하기위한 전송측번지의 지정

DI:데스티네이션 인덱스 - 번지의 간접번지에 사용 , 특히 스트링명령에 있어서는
레지스터로 부터 메모리에 데이터를 전송하기위한 수신측 번
지를 지정할때 사용

세그먼트의 개념

8086은 1M 바이트 까지의 메모리를 취급할수가 있음니다.
1M바이트는 2의 20승 이므로 번지데이터로서 20비트가 필요하게 되는것입니다.
8086레지스터는 16비트 크기 밖에 없으므로 2개의 레지스터를 조합시켜 20비트의 번지를 나타내는 방식을 취한다.
예:

```
  2000 H     >세그먼트레지스터
 + 3456H     > 오프셋   IP 나  BX, SI (데이터를 읽고 쓸때)
  ------
  23456H       오프셋이란 : 어떤 기준 번지로부터의 변위(차이)를 나타낸다.
               오프셋 번지는 BX,SI,IP 등의 레지스터에 의해 나타내는것 외에
               직접 수치로 지정될수도 있읍니다.
                          0H              메모리
세그먼트 레지스터      >20000H   >

      오프셋 번지      > 3456H                >23456H
                                               물리번지
```

오프셋값만을 지정하면 자동적으로 세그먼트 레지스터의 값은 더하여 계산된다.
따라서 세그먼트 레지스터의 값을 한번 설정해놓으면 ,세그먼트 베이스로부터 64KB 이내의 번지는 오프셋 번지를 지정한느것만으로 표시할수있다.

세그먼트 레지스터의 용도

CPU 가 명령을 읽어낼때:
CS:IP 의 값으로 부터 물리번지 를 생성 ,그번지로 부터 명령을 꺼낸다.

데이터를 전송하는경우:
DS: 와 OFFSET (오프셋) 번지 로 부터 데이터가 저장되어있는 물리번지를 계산

스택 동작을 수행할때:
SS:SP 로부터 스택동작을 수행하는 번지를 계산

세그먼트는 서로 전부 혹은 일부가 중복되어도 상관이 없으므로 필요 하다면 몇개의 세그먼트를 같은 물리 번지에 배치 할수도 있음니다.

명령의 개략적인 해설

데이터 전송 명령: MOV
사칙연산 명령 : ADD: 덧셈, ADC:덧셈, SUB:뺄셈, SBB:뺄셈
MUL: 곱셈, IMUL:부호달린 곱셈,
DIV: 나눗셈, IDIV:부호달린 나눗셈
CBW: 바이트에서 워드로 부호확장
CWD: 워드에서 더블워드로 부호확장
INC: 하나 증가
DEC: 하나 감소
논리연산 .쉬프트명령:AND: 논리곱, OR: 논리합,
XOR: 배타적 논리합, NOT:부정 ,NEG: 부호반전
SHL: S는 shift ,H는 0을 넣을 것인가 , L 은 left
ROR: R은 Rotate,
비교분기 명령: CMP, JMP는 무조건 분기,
Above(크다),Below(작다.) ,Greater(부호를 포함해서 크다.)
Less(부호를 포함해서 작다) ,Equal(같다), Not(부정)
LOOP: 반복
LOOPE: loop if equal 조건부 반복
LOOPNE: loop if not equal
CALL: 서브루틴으로 분기
RET: 서브루틴으로 부터 원래의 루틴으로 돌아올 때에
스트링 명령: LODS: 메모리로 부터 레지스터에 데이터를 로드
STOS: 메모리에 데이터를 저장하는 명령
LODS, STOS는 메모리의 번지지정 방법이 SI 혹은 DI레지스터를 사용하여 간접지정으로 정해지기 때문에 미리 SI,DI에 번지를 세트해 두어야 한다.
LODSB,STOSW,MOVSB,MOVSW: 블럭전송
단독으로 1바이트,1워드의 데이터를 전송
RET(repeat) 명령과 조합시켜 사용 하면 cx 레지스터가 지정하는 횟수만큼 반복하여 데이터를 전송합니다.
이때 번지는 자동으로 갱신되어가므로 한 명령으로 연속된 여러 데이터를 전송할수가 있읍니다.

스트링 명령에는 그 밖에도 데이터의 전송은 하지 않고 레지스터와 메모리의 내용을
비교만 하는 SCAS(scan string),
메모리 끼리의 내용을 비교하는 CMPS(compare string)
이들 명령은 REPE(repeat until equal)
REPNE(repeat until not equal)명령과 조합함으로써 일치하는
데이터가 얻어질 때까지 ,혹은 일치하지 않는 데이터가 얻어질
때 까지,메모리상의 데이터를 탐색할수가 있읍니다.
SCASB
REPE SCASW
REPNE CMPSB
REPE CMPSW

I/O 명령: LSI에 명령을 보낸다든지 데이터를 얻는다든지 하기위한 명령이 I/O 명령
I/O 포트에 데이터를 보내는 명령이 OUT
I/O 포트에 데이터를 얻는 명령이 IN
포트의 번호는 직접 수치 혹은 DX 레지스터를 사용하여지정
데이터는 AX 혹은 AL 레지스터를 이용하여 전송

인터럽트 명령: INT (interrrupt) 다음에 번호를 지정
인터럽트 처리 루틴으로 부터 원래의 루틴으로 돌아 오려면
IRET(interrupt return)을 사용

CPU 제어명령: 주로 8086의 cpu가 수치연산 프로세서 8087과의 사이에서 데이타를 전
송한다든지 주변장치로 부터 READY 신호가 올때까지 실행을 정
지하고 기다린다든지 하기위한 명령입니다.
WAIT,ESC,LOCK,HLT : 외부 주기
MOT: 아무수행도 하지 않음

그 밖의 명령 : 레지스터 혹은 메모리의 내용을 스택영역으로 대피 복귀시키기 위한
PUSH ,POP ,
플래그 레지스터를 스택으로 대피 복귀하는 PUSHF,POPF
두개의 레지스터 혹은 메모리의 데이터를 교환하는 XCHG
한 바이트씩 나열된 데이터의 N 번재 의 것을 꺼내는 XLAT

플래그 레지스터를 직접 제어하는 :
STC(set carry flag),CLC(clear carry flag),
CMC(complement carry flag),
STD(set direction flag) ,CLD(clear direction flag),
STI(set interrupt-enable flag)
CLI(clear interrupt-enable flag)
LAHF(load AH from flags),SAHF(store AH to flags) :
플래그레지스터 하위 8바이트와 AH 레지스터의 데이터를 전송

AAA(ASCII adjust for addition),
DAA(decimal adjust for addition),
AAS(ASCII adjust for subtract),
DAS(decimal adjust for subtract),
AAM(ASCII adjust for multiply),
AAD(ASCII adjust for division) :
플래그 레지스터 하위 8바이트와 AH 레지스터의 데이터를 전송
(이상 모두 오퍼랜드를 갖지 않음)
LEA(load effective address):실효번지를 로드
LDS(load data segment register),
LES(load extra segment register):
세그먼트 레지스터를 포함 하는 실효번지를 로드함

명령의 구성

```
L1:     MOV     AX,BX   ;comment
```

 라벨   작동 코드   제 1 오퍼랜드   제 2 오퍼랜드   설명문

```
                        ^
```

조작의 방향

명령에 대해서는
제 2오퍼랜드로 부터 제 1 오퍼랜드의 방향으로 조작이 이루어 집니다.

L1: 과 같은 명령은 직접적으로 는 기계어 코드로 번역되지 않고 ,분기명령등에서 참조 될때에.번지의 계산에 사용됩니다.
이와 같은 명령을 의사 명령이라고 말하고 ,어셈블리 프로그램을 작성하는데 없어서는 안되는 것입니다.

의사명령이란 무엇인가.?

어셈블러가 소스(source) 프로그래을 기계어 코드로 변환할때에는 필요한 지시를 어셈블러에게 행하는것 입니다.

장소(번지)를 지정하는 라벨

```
 L1:    MOV  AX,BX
        .............
        .............
        JMP   L1        > L1 번지로 무조건 뛰라는 명령
```

변수이름은 메모리상의 번지를 지정한다.

```
        MOV  AX,DATA1     > DATA1 번지에 있는 메모리 워드1234H 를 AX 에 전송하는
.......                     간접번지 지정방식으로 데이터 전송
.......
DATA1   DW   1234H
```

>define word 란 변수정의 의사명령

PTR - 데이터의 타입을 나타낸다.

```
BYTE      PTR
```

SEGMENT - 세그먼트 이름을 선언

```
MAIN    SEGMENT PARA PUBLIC 'CODE'
```

PUBLIC - 외부 참조 가능임을 나타낸다.

```
PUBLIC  PROC1
```

IF - 조건어셈블을 지정

```
IF      IBMPC EQ  TRUE
.........
ENDIF
```

의사명령에는 절대 필요한것과 그렇지 않은 것이 있다.

시스템 호출이란 무엇인가 ?

입출력을 위한 시스템 호출(system call):
시스템에 준비되어 있는 기본 루틴을 사용하기 위한 방법
인터럽트와 시스템 호출은 거의 비슷하다.
MS-DOS 에서는 주로 인터럽트의 21H 번을 사용 ,그중에서 서브커멘드(subcommand)
를 사용한다.

시스템호출을 사용하면 프로그램을 효율적으로 개발을 할수가 있다.
그러나.주의 사항도 있음니다.
1)다른 OS 상으로 이식을 할때에는 그대로 금방 이식할수가 없다
2)입출력외의 루틴은 1)독자적으로 작성하든지 2)ROM 내의 루틴을 이용
    전자의 경우 프로그램개발에 방대한 시간을 걸린다.
    후자의경우 타기종과의 호환성이 사라진다는 문제 점이 있읍니다.
그래서 이부분은 어셈블러에 의한 프로그램의 개발에 있어서 큰 문제 입니다.
(실수연산을 고급언어에 맡겨 링크 하는 방법등을 생각할수있다.)

이러한 문제로 시판되는 소프트웨어 중에서는 직접 BASIC ROM의 루틴 등을 사용하고 있는것이나 특수한 인터럽트 처리를 사용 하고있는것도 있어서 MS-DOS 상에서는 뜨지만 IBM-PC 이외의 기종에서는 동작하지 않는것도 있다.

기능의 발달과 범용성의 확대라는 두가지 방향에는 모순되는 측면이있다.

어셈블러의 사용법

오브젝트 화일 이란 : 어셈블의 결과 출력되는 기계어의 중간화일
소스 리스팅(source listing): 소스 화일 과 어셈블의 결과 출력된 기계어를 대응시킨
                리스트 화일
크로스 레퍼런스(cross reference): 라벨이나 변수 이름 이 여기서 정의 되어 어디에
                서 참조 되고 있는가를 리스트하는 화일 입니다.

링크의 실행

링커의 역활은 몇개의 오브젝트 화일을 합쳐서 하나의 프로그램으로 한다든지,
라이브러리로 부터 필요한 루틴을 꺼내어서 프로그램에 결합하는 것에 있읍니다.

오브젝트 모듈 (object module) - 어셈블의 결과 출력되는 오브젝트 화일
다른 모듈과 결합 - + 기호를 사용
실행화일 (run file) - 링커에 의해 작성된 실행가능 화일
리스트 화일 - 실행가능 화일 중에서 세그먼트나 프로시듀어의 번지나 길이를 나타냄

링크의 결과 다음과 같은 메시지가 나오고 링크가 끝납니다.
Warning: no STACK segment
There was 1 error detected

스택세그먼트 는 설정하지 않아도 OS 가 자동적으로 설정해 주는 것으로 되어있으므로 그대로 실행할수가 있읍니다. (다른 에러 를 무시 할수는 없음니다.)
COM 화일은 EXE 화일을 변환 하여 만듭니다.
EXE2BIN 을 이용 확장자가 .BIN 을 만들고 , 확장자명인 BIN 을 변경하면됩니다
EXE 와 COM 화일중 COM 화일이 먼저 실행된다.
스택 세그먼트가 선언 되어 있는 EXE 화일은 변환할수 없다.
COM 모델과 EXE 모델과는 세그먼트의 초기 설정 등에 차이가 있다

디버그의 사용법

디버그는 프로그램을 조금씩 실행시키면서 레지스터나 메모리의 내용을 살펴 보는 것으로서, 프로그램이 기대한 대로 동작하고 있는가를 살피기 위한 TOOL 입니다.
본질적인 알고리즘의 잘못을 찾아내는 것은 쉽지 않지만 어느 부분이 폭주해 버리는가 라는 것은 알수 있읍니다.

EXE 모델과 COM 모델

EXE 모델은 비교적 큰 프로그램에 COM 은 모델이 비교적 작은 프로그램에 사용 된다
MS-DOS 에서는 실행가능 프로그램이 메모리 상에 로드 되었을 때에 프로그램의 직전에 PSP(Program Segment Prefix)라고 부르는 부분이 설정되고,그 다음에 프로그램 본체가 놓여집니다. PSP 는 전부 100H 바이트가 있어서 프로그램 실행에 필요한 여러가지 정보가 들어있습니다.

EXE 모델
DS 와 ES 가 PSP 의 시작을
CS 가 프로그램의 시작을 나타내도록 세트됩니다.
SS 는 특별히 지정하지 않으면 CS 와 같은 값이 됩니다.
프로그램의 실행은
CS 내의 IP 로 나타내는 번지 부터 시작됩니다.
시작번지는 특별히 지정하지 않으면 0
임의의 번지 부터 시작할려면 (소스 프로그램의 END 문으로 지정한다.)
(주의)
DS와 ES 가 PSP 의 부분을 지정하고 있으므로 프로그램 중에서 DS 및 ES 를 사용할때는 반드시 자기가 설정을 다시 하지 않으면 안된다.

*.EXE 모델의 세그먼트의 초기치

```
    하위번지                    >DS,ES
              PSP100바이트
                                 >CS,(SS)
              프로그램           >CS:IP(시작번지)
    상위번지
```

COM 모델
COM 화일 을 로드 하였을때 에는 CS,DS,ES,SS 의 4개가 모두 일치하여
PSP 의 시작을 지시 하도록 세트됩니다.
프로그램의 본체는 100H 바이트의 PSP 직후에 놓여지고 선두로 부터 실행,
IP 의 초기치는 100H 로 세트 됩니다.
소스프로그램을 작성할때에도 ORG 명령을 사용하여 시작번지를 100H 번지부터설정
동시에 프로그램의 시작에 라벨을 붙여서 END문 에 의해서 시작번지를 지정하지 않으면 않됩니다.
또한 COM 화일 에서는 세그먼트의 값을 변경하면 안됩니다.

*.COM 모델의 세그먼트의 초기치
하위번지
>CS,DS,ES,SS
PSP100바이트
>CS:IP(=100H)
프로그램 시작번지
상위번지

디버그의 실제

A>DEBUG SAMPLE.EXE
_ >디버거의 프롬프트
레지스터 내용을 표시하는 R 커멘트
역어셈블 U 커멘트
실행을 하는 G 커멘트
실행 커멘드(GO)의 시작번지는 = 을 사용 _ G=0,F
디버거를 끝내는 Q
다음 메세지가 나왔을때 한번더 실행을 하려고 하면 폭주해 버린다.
Program terminated normally
의 메세지가 나오면 일단 디버거를 마친다음 에 다시 처음부터 시작해야 한다.

프로그램의 수정법

디버거를 기동후 화일 을 로드 하는 법: N 커멘드
화일을 로드하는 : L 커멘드
_NSAMPLE.EXE
_L
1스텝씩 실행하는 : T 커멘드
_T=100,20 20은 20 스텝
T 커멘드는 서브루틴이나 인터럽트 처리 루틴의 내부까지 실행
그래서 인터럽트 처리 루틴이나 서브루틴을 만났으면
그것을 건너 뛸필요가 있음니다.
서브루틴을 건너 뛰려면 : G 커멘드
_G12F
1행을 어셈블 : A 커멘드
_A8
2242:0008 CMP DL,5F
2242:000B
메모리 내용을 직접 바꿔 써 넣으려면 대치(substitute)명령인: S 커멘드
단 한행씩 바꿔쓴 앞뒤의 바이트수가 변화되면 파괴됩니다.
수정한 화일을 원래의 디스켓에 써넣으려면 기록(WRITE)명령인: W 커멘드
단 실행가능 화일인 EXE 나 COM 화일은 바꾸어 써넣을수가 없읍니다.
이것은 디버거가 화일을 메모리상에 로드 할때에 특별한 처리를 하여 실행이 가능한
상태로 만들어 놓고 있기 때문입니다.
보통은 디스크상의 이미지 (image) 가 그대로 메모리에 상에 로드 되기 때문에 수정하
여 다시 써넣을수가 있지만 , 이들화일은 디스크상의 이미지와 메모리 상의 이미지가
차이가 있으므로 불가능한 것입니다.