본문 링크 : <https://www.hackerschool.org/HS_Boards/zboard.php?id=Free_Lectures&no=1133>

어셈블리 총 정리 강좌   
  
    
  
아직도 어셈블리언어가 필요한가.  
--------------------------------------------------------------------------------  
어셈블리  언어의 지식은 고속연산 루틴을 작성하기위해서만 필요한 것이 아니라.마이  
크로 컴퓨터의 시스템을 잘알기 위하여 필요한것이다.  
메모리상의 데이터나 I/O 기기를 직접 액세스 하는등, 고급언어에서는 할수없는 조작  
을 할 수 있다 라는 것이다.  
CPU 가 이해할 수 있는 것은 수치로 기술된 명령 밖에 이해 할 수가 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 작성해주는프로    
그램.  
MASM 은 모듈별 개발이 용이하므로 축적된 소프트웨어를 유효하게 활용 할수있다.  
  
MASM 에 의한 어셈블러 개발법 : 아스키 문자열로써 저장하는 형태의 에디터라면 무엇  
이든 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 오브  
젝트 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 섦계되어 있으므로 효율이 좋은   
프로그램을 작성 할때는 이것을  고려 해볼 필요가 있다.  
특히 ,데이터를 액세스 할때는 0 번지와 1번지  2번지와 3번지 이러한 식으로 짝수 홀  
수 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 번  
지를 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 이  
내의 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 지정  
                     하는 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 어셈  
블러에게 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 비숫하다.  
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 뜨지  
만 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 것  
으로서, 프로그램이 기대한 대로 동작하고 있는가를 살피기 위한 TOOL  입니다.  
본질적인 알고리즘의 잘못을 찾아내는 것은 쉽지 않지만 어느 부분이 폭주해 버리는   
가 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 사용  
할때는 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 않  
으면 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.  
  
MOV 명령과 문자 출력  
데이터의 입출력과 전송명령에 대해서 해설 합니다.  
\*.레지스터에 수치를 대입한다.  
\*.레지스터와 레지스터 사이에서 데이터를 전송한다.  
\*.레지스터와 메모리 사이에서 데이터를 전송한다.  
  
화면에 " AB " 를 표시 하는 프로그램  
MAIN    SEGMENT             ; 세그먼트를 알리는 의사명령  
        ASSUME  CS:MAIN     ;  
;  
        MOV     DL,41H      ; 아스키코드 41H ' A  ' 자 이다.  
        MOV     AH,2  
        INT     21H  
        MOV     DL,'B'  
        MOV     BL,2  
        MOV     AH,BL  
        INT     21H  
;                            ; 단순히 줄을 띄우기 위해 삽입된 설명문  
        MOV     AH,4CH       ; 프로그램의 끝냄  
        INT     21H  
;  
MAIN    ENDS  
        END  
  
자기가 작성한 소스프로그램을 그대로 보고 싶으면 리스트 화일인 .LST 를 살펴  
보는 것이 좋다.의사명령은 직접 기계어로 번역되지 않지만 , 어셈블 작업의 흐름을  
정할때에 중요한 역할을 해 줍니다.  
여기서 MAIN 은 어떤 이름이든지 상관없다.  
ASSUME  CS:MAIN  
이라는 의사 명령은 어셈블러가 어셈블을 할때 코드세그먼트(CS:)를 참조 하라는 명령  
이 있으면 MAIN 이라는 이름의 세그먼트를 참조하라는 선언 을 하는 문장입니다.  
ASSUME 문장은 4개의 세그먼트(CS,DS,SS,ES)와 세그먼트 이름을 대응시키는  선언문입  
니다.  
명령은 반드시 CS:(코드세그먼트 )에  
END 문으로 어셈블러는 끝난다. END 문으로 시작번지를 지정할수가있다.  
  
  
어셈블러 프로그램의 기본형  
AMIN    SEGMENT  
        ASSUME  CS:MAIN  
       ..........  
        본 문  
       ..........  
MAIN    ENDS  
        END  
  
  
설명문 (comment,  주석문)을 쓰는법  
;  
은 설명문 행 입니다. ; 이 다음 부터 쓰여진 문자는 설명문입니다.  
  
  
8086 의 레지스터(register)  
--------------------------------------------------------------------------------  
범용 레지스터 AX,BX,CX,DX  는 8 비트로 나누어 사용할수 있다.  
범용 레지스터 SI,DI,BP,SP  는 16 비트로서 밖에 사용 할수 없다.  
  
어셈블러에서는 특별히 표시하지 않는한 수치는 10진수로서 취급됩니다.  
16진수를 사용할때는 숫자의 뒤에 H(Hexa decimal) 를 붙여서 표시합니다.  
수치가 알파벳으로 시작할때는 " 0 "을 붙여 변수이름과 혼동을 피한다.  
  
MOV     DL,  41H                ; DL  <----- 41H      41H를 DL 에 대입하라.  
   +--- |    | ----------+  
   |    +----+           |  
행선지(DESTINATION), 출원지(SOURCE)  
  
시스템 호출과 입출력은   INT  21H <---- 인터럽트 (interrupt-끼어들기) 명령은  
강제로 CPU가 하던 일을 중지시키고 끼어들기  
  
MOV     DL,'1'      ; 문자를 인용부호로 에워싼 것을 오른쪽에 쓴경우에는  
                      그문자에 대응하는 아스키코드가 쓰여진 것으로 간주됩니다.  
                      1 은 아스키코드값 31H 로 대치됩니다.  
  
프로그램을 끝마치는 방법  
MOV     AH,4CH     ; MS-DOS 의 SYSTEM CALL 인 4CH 번째의 루틴  
INT     21H        ; 프로그램을 마치고 OS 레벨로 돌아간다.  
  
  
데이터를 두는 방법(1)  
--------------------------------------------------------------------------------  
프로그램중에 데이터를 두고 , 그 데이터를 레지스터에 전송하는 방법을 설명  
변수이름 선언법 :  
XXX     DB      'X'       ; DB 는 define byte  
YYY     DW      3456H     ; DW 는 define word  
|        |        ---------> 변수에 들어 가는 초기치  
|        +---------------->변수의 형  
+----------------------->변수이름  
DB ..... 바이트의 정의  
DW ..... 워드(2바이트)정의  
DD ..... 더블 워드(4바이트)정의  
DQ ..... 쿼트 워드(8바이트)정의  
DT ..... 10 바이트 정의  
\*.사용법  +----------+--------+-----------+  
          | 변수이름 |   DB   |   식      |  
          +----------+--------+-----------+  
  
세그먼트를 정의 하는 ASSUME 의사 명령  
MAIN    SEGMENT  
        ASSUME  CS:MAIN, DS:MAIN  
MAIN :     세그먼트 개시를 선언  
CS:MAIN    코드세그먼트가 MAIN 이라는 이름의 세그먼트에 연결되어있다는것  
DS:MAIN    데이타세그먼트가 MAIN 이라는 이름의 세그먼트에 연결되어 있다는것  
  
8086에서는 데이타는 데이타세그먼트 내에 있는 것으로서 해석되므로 ,데이터세그먼트  
가 어디있는가 를 지정해야한다.  
  
명령도 데이타도 둘다 MAIN 이라는 이름의 세그먼트 내에 있으므로  
ASSUME  CS:MAIN, DS:MAIN  
  
그러나 이것은 어셈블러에 대한 지시이다  
데이타를 처리하는 명령으로는 번역되지않는다  
세그먼트를 초기설정(initialize) 하기위해서는  
MOV     AX,CS  
MOV     DS,AX  
이와같이하여 ,AX 레지스터를 경유하여 DS 에  CS 의값을 전송하여  
2 개의 세그먼트를 일치시키지않으면 안됩니다.  
이것은 어셈블러의 정해진 문구입니다.  
여기서  
MOV     DS,CS  
라고 할수는 없읍니다.세그먼트 레지스터(CS,DS,SS,ES)는 범용 레지스터들로 밖에는  
데이터를 전송하는 것이 허용되어 있지않기 때문입니다.  
세그먼트 레지스터에 직접 숫자나 변수를 대입할수는 없읍니다.  
어셈블 리스트에서  
8A 16 0020 R  
R  .... 이라는 문구는 상대번지(relative address) 지정방식임을 나타내고 있읍니다.  
  
  
데이터를 두는 방법(2)  
--------------------------------------------------------------------------------  
여러개의 세그먼트에 두는 법  
MAIN    SEGMENT  ---+  
        ......      |  
MAIN    ENDS     ---+  
DATA    SEGMENT  ---+  
        .....       |  
DATA    ENDS     ---+ 이와 같은 두개의 세그먼트가 존재하게 설계한다.  
  
ASSUME  CS:MAIN,DS:DATA  
데이터를 참조하는 명령이 있는 경우에는 데이터는 DATA 라는 이름의 세그먼트에 있다  
고 간주하고 명령을 작성하라는 지시입니다.  
  
MOV     AX,DATA  
MOV     DS,AX  
데이터의 세그먼트의 값을 초기 설정하는 부분입니다.  
  
프로그램을 시작할때 데이터 세그먼트의 지정법만 잘 익혀두면 본분 중에서는 데이터  
가 어느 세그먼트 내에 있는가  신경쓰지 않고 ,프로그램을 작성할수있읍니다.  
  
  
번지의 간접 지정법(1)  
--------------------------------------------------------------------------------  
바이트형 데이터는 연속하여 정의할수있다.  
DB 를 사용하여 데이터를 정의 하는 경우에 ' ABC ' 와 같이 연속하여 몇 문자라도  
데이터를 정의 할수가 있읍니다.  
XXX     DB      'ABC'      ------+ 모두같다.  
XXX     DB      'A','B','C'      |  
XXX     DB      41H,42H,43H   ---+  
        |  
        +---->DW,DD,DQ,DT 와 같은 다른 변수형의 데이터도 나열할수는 있습니다.  
그러나 수치로써는 가능하지만 문자열로써 주어지는 경우는 한번에 2문자 씩의 문  
자열밖에 줄수가 없다.  
AAA     DD      'AB','CD','EF'는 허용이 되지만  
AAA     DD      'ABCD', 'EFGH' 는 허용이 안된다.  
  
DB 의 경우에만이 특별하고 3 문자 이상의 문자열의 나열을  
'A','B','C'    로 하는 대신에     'ABC' 로 할수있다.  
  
자,그럼 어떻게 XXX 라는 변수이름의 장소에 서 나열되어있는 데이터를 꺼내어 쓸수가  
있는가  
MOV     AL,XXX  
라고 하면 XXX 의 장소에 나열된 문자열의 제일 처음 의 것인  'A'=41H 가 AL 레지스  
터에 전송됩니다.(XXX라고 하는 이름의 장소에 있는 데이터를 AL레지스터에 전송하라)  
  
변수가 자리잡고 있는 번지를 꺼내는 오프셋(OFFSET 명령):  
MOV     BX,offset XXX  
                  |      변수이름이 붙여진 번지 자체의 값을 BX 레지스터에 전송하  
                  +-----> 는 명령입니다.  
OFFSET :  변수이름이 붙여진 데이터가 있는 번지를 꺼내는 연산자.  
오프셋번지 : 세그먼트를 기준으로 한 번지  
  
MOV     BL,XXX ; BL 과  XXX 는 8 비트 이므로 가능  
MOV     BX,XXX ; BX 는 16비트 레지스터, XXX는 8비트 레지스터이므로 에러  
  
[  ] 를 사용한 간접 번지 지정방식:  
어떤번지의 내용을 꺼낼때에는 [번지 ] 와 같이 [  ] 를 붙인다.  
MOV     DL,[1FH] ; 1FH 번지에 있는 데이터를 DL 레지스터로 전송하라  
  
BX 레지스터내에 변수 XXX 의 처음 데이터가 DL 레지스터로 전송됩니다.  
그 다음은 이것을 표시 합니다.  
2 번째 데이터는 선두 번지 +1 번지에 저장되어 있으므로, 여기서는 BX + 1 번지의  
내용을 꺼내면 되는 것입니다.같은 방법으로 BX+2,BX+3 .......  
MOV     DL,[BX+1]  
[  ] 내의 값이 나타내는 번지에  
저장되어 있는 내용을 꺼내는 것을 지시하고 있읍니다.  
이와 같은 지정법을 간접지정방식(indirect addressing )이라고 부릅니다.  
MOV     AX,[1234H]  
와 같은 지정법도 간접번지 지정방식의 일종으로서의  
직접번지지정방식(direct addressing mode) 이라고 부릅니다  
MAIN    SEGMAIN                ;  \*.데이터를 세그먼트 내에두는 경우  
        ASSUME  CS:CODE,DS:DATA  
  
        MOV     AX,DATA  
        MOV     DS,AX  
        +-----------+  
        | 본  문    |  
        +-----------+  
MAIN    ENDS  
  
DATA    SEGMENT  
XXX     DB      'ABC'  
DATA    ENDS  
        END  
  
번지의 간접 지정법(2)  
--------------------------------------------------------------------------------  
간접지정에 사용할수 있는 레지스터는 4개뿐이다  
BX,BP,SI,DI  
왜그런가 하는 것은 8086 설계자가 정해놓은 일이다.  
  
[BX+DI]  
[BX+SI+1]  
와 같은 이들끼리의 조합지정이 가능 합니다  
예) [BX+SI+4]  
BX=1000H  
SI=1234H  인 경우  
1000H + 1234H + 4H = 2238H 이 된다.  
  
간접지정 조합법의 규칙:  
+----+       +-------+       +--------------+  
| BX |       |  SI   |       |  16비트 숫자 |  
| BP |   +   |  DI   |    +  |  8비트 숫자  |  
|없다|       |  없다 |       |  없다        |  
+----+       +-------+       +--------------+  
  [BP] 는 단독으로 사용할수가 없음니다.  
단순히 기계적 제약에서 오는 것입니다 사용하고 싶다면   [BP+0] 으로 합니다.  
MOV     DL,[BX+SI]  
; BX에는 변수의 시작번지 , SI에는 3을 대입,  
따라서 DL 에는 변수의 선두로 부터 4 번째 바이트의 데이터가 DL 레지스터에  전송한  
다 .   \*.SI= 0 부터 선두를 가리킨다.  
  
예제) MOV5.ASM  
메모리에서 저장된 문자열" ABCDEFG" 중 네번째, 다섯번째 문자 DE 를 출력  
MAIN    SEGMENT  
        ASSDUME CS:MAIN,DS:DATA  
;  
        MOV     AX,DATA  
        MOV     DS,AX  
        MOV     BX,OFFSET AAA  
        MOV     SI,3  
        MOV     DL,[BX+SI]  
        MOV     AH,2  
        INT     21H  
        MOV     DL,[BX+SI+1]  
        MOV     AH,2  
        INT     21H  
;  
MAIN    ENDS  
;  
DATA    SEGMENT  
AAA     DB      'ABCDEFG'  
;  
DATA    ENDS  
        END  
\*. 간접지정에 사용 하는 레지스터에    BP  레지스터가 포함되는 경우에는 ,  
   스텍세그먼트(SS:)에 데이터가 있는 거으로서 해석된다.  
(  보통은 BP 레지스터를 사용한 간접 지정법은 안하는것이 무난할 것입니다. )  
  
  
간접 번지 지정방식의 사용법  
--------------------------------------------------------------------------------  
CODE    SEGMENT  
        ASSUME  CS:CODE,DS:DATA  
;  
        MOV     AX,DATA          ; DS를 설정  
        MOV     DS,AX  
;  
        MOV     BX,OFFSET AAA    ; AAA의 번지값을 BX 에 대입  
        MOV     AX,'AB'          ; 문자 하나는 8비트  
        MOV     [BX],AX          ; BX 가 가리키는 AAA 변수에 'AB'를 넣는다  
        MOV     CX,AAA  
        MOV     DL,CH            ; DL 에 A 를 넣는다  
        MOV     AH,2       ----+  
        INT     21H        ----+ 문자 A를 보인다.  
        MOV     DL,CL            ; DL 에 B 를 넣는다.  
        MOV     AH,2       ----+ 문자 B를 보인다.  
        INT     21H        ----+  
        MOV     AH,4CH  
        INT     21H              ; 종료하고 OS 로돌아간다.  
;  
CODE    ENDS  
;  
DATA    SEGMENT  
AAA     DW                     ; AAA 라는 변수를 위해서 워드 영역을 확보하라  
DATA    ENDS  
        END  
  
AAA     DW      ?  는   '? ' 는 숫자는 아무것이라도 좋다라는 의미입니다.  
  
이밖의 사용법으로는  
MOV     [BX+SI+5],1234H  
MOV     AAA,'AB'  
라는 사용법이 가능 합니다.  
  
MOV     [BX],[SI+3]  
그러나  이와 같은 ,오퍼랜드 양쪽 다 간접지정으로 할수는 없읍니다.  
  
MOV     BX,[0008] -----> 8 번지의 내용을 꺼내온다 라는 간접 지정방식  
레지스터를 사용하지않고 수치만으로 번지를 지정하는 이러한 방법을 직접번지 지정  
(direct addressing)이라고 말합니다.  
또한, MASM 에서는 이와 같은 직접 번지 지정방식을 어셈블러 속에 쓸수가 없읍니다.  
그것은 메모리 번지는 링커에  의해 동적(dynamic)으로 활당되기때문에 ,고정된  번지   
지정방식을 사용 할수없다고 가정되기 때문이라고 생각됩니다.  
  
EQU 의사 명령  
--------------------------------------------------------------------------------  
상수를 정의 하는 EQU 의사 명령  
MAIN    SEGMENT  
        ASSUME  CS:MAIN  
CHAR1   EQU     'A'    ;'A' = 41H 가 정의 됩니다 --+  
CHAR2   =       42H    ;  = 는 재정의가 가능       |  
DISP    EQU     2      ;  EQU는 재정의가 불가능, --+상수는 어셈블할때 치환  
;  
        MOV     DL,CHAR1  
        MOV     AH,DISP  
        INT     21H  
        MOV     DL,CHAR2  
        MOV     AH,DISP  
        INT     21H  
        MOV     AH,4CH  
        INT     21H  
;  
MAIN    ENDS  
        END  
  
  
EQU 의사명령에 의한 상수의 정의와 DB 등에 의한 변수의 정의와의 차이는 :  
EQU 에 의해서 선언된 상수 - 어셈블할때 직접숫자로 치환  
DB 등에 의해서 선언된 변수 - 데이터가 저장되어 있는 번지로서 치환  
예)  
MOV CHAR1,DL  ----->  불가능  
'=' 에 의해서 정의된 숫자를 다시 정의 할때는 ,다시 '=' 를 사용해서 정의 한다.  
어셈블러는 시작순서대로 어셈블을 해나가면서 가장 최근 에 정의된 숫자를 그 상수  
의 숫자로써 사용합니다.  
따라서 다음과 같이 됩니다.  
CHAR1   =       41H  
        MOV     DL,CHAR1  
CHAR1   =       56H  
        MOV     DL,CHAR1  
결과 :  
=       0041  
0000    B2      41  
=       0056  
0002    B2      56  
  
  
데이터의 형과 PTR 연산자  
--------------------------------------------------------------------------------  
데이터의 형(type):  
바이트,워드(2바이트),더블워드(4바이트)  
데이터의 형은 레지스터의 크기와 같다고 간주된다.  
  
주의 )  
오퍼랜드에 레지스터를 포함하지 않은 경우 크기의 지정이 필요  
예) 한쪽이 간접 번지지정 ,또 한쪽이 숫자의 경우입니다.  
  
MOV     [BX],12H  
여기에서 BX 의 값이 10H 라고 합시다.그런데 여기서는  
1) 10H 번지에 바이트 데이터 12H 를 저정할것인지,  
2) 10H 번지와 11 번지에 워드 데이터 0012H  를 저장하는 것인지 알수가 없음니다.  
  
\*.바이트 데이터의 경우                       \*.워드데이터의 경우  
번지  |        |        |         |          번지 |         |         |       |  
      +--------+        +---------+               +---------+         +-------+  
10H  | 55     |  12H   |  12     |           10H |   55    | 0012H   |  12   |  
      +--------+------> +---------+               +---------+-------> +-------+  
11H  | 66     |        |  66     |           11H |   66    |         |  00   |  
      +--------+        +---------|               +---------+         +-------+  
      |        |        |         |               |         |         |       |  
12H 를 바이트 데이터로써 저장한 경우에는 11H 번지의 내용은 사라지지 않고 남지만  
12H 를  워드 데이터로써 저장한 경우에는  11H 의 내용은 지워져버립니다.  
  
  
  
PTR 연사자의 사용  
--------------------------------------------------------------------------------  
데이터의 형을 명확하게 하기위하여  
MOV     BYTE PTR [BX],12H  ; PTR 은 POINTER 의 약자  
MOV     WORD PTR [BX],12H  
이와 같이 전송되는 측에 BYTE 또는 WORD 를 지정합니다.  
이때 PTR 이라는 오버라이트 (overwrite) 연산자와 함께 사용하도록 되어 있읍니다.  
변수로 선언되어 있는 형이 DB 이든 DW 이든  BYTE , WORD 양쪽다 사용할수가 있읍  
니다.  
  
MOV 명령과 번지지정방식의 정리  
---------------------------------  
ADDRESING MODE:  
\*.즉치 방식(immediate mode)  
AAA     EQU     1234H  
        MOV     AX,AAA  ;메모리의 바이트수가 레지스터바이트 이내일것  
CS,DS,ES,SS 및 IP,FL 레지스터에 직접 수치를 전송할수는 없읍니다.  
  
\*.직접 방식(레지스터 번지지정 방식)  
레지스터의 내용을 직접 전송하는 방식  
MOV     DS,AX  
MOV     CL,BH   ; 양쪽의 레지스터의 크기가 일치하는 8비트 혹은 16비트여야 한다.  
세그먼트 레지스터간의 전송은 할수가 없다 . IP ,FL 레지스터는 사용할수가 없고 ,  
CS 레지스터로의 값을 전송은 할수 없읍니다.  
  
\*.간접 방식(메모리 번지지정 방식)  
전송하는 값이 저장되어 있는 번지를 지정- 방법2가지  
1) BX,BP,SI,DI 의 4 가지 레지스터와 숫자를  조합시킨다.  
MOV     AX,[BX+DI+4]  
2) 데이터를 변수로서 정의하고 ,그변수이름을 사용하여 지정하는 경우  
MOV     AX,AAA ; 숫자가 있는 번지에 붙여진 이름을사용하여 간접지정을 하는것  
  
간접 방식에서의 번지지정법:  
+-  BX  -+   +-  SI  -+   +-   8비트 숫자  -+  
|   BP   | + |   DI   | + |   16비트 숫자   |  
+- 없음 -+   +- 없음 -+   +-    없 음      -+  
  
위에서 허용되지 않는 조합방법  
1) 8비트 숫자만을 사용하는 경우( 번지는 16비트가 아니면 지정할수 없다 )  
그러나  MOV   AX,[1234H] 와 같은 직접 번지지정 방식은 사용할수 없다.  
직접 메모리 번지를 지정하여 간접 방식을 사용하고 싶을때는 -  
MOV     BX,1234H  
MOV     AX,[BX]    ; 와 같이 하지 않으면 안됩니다.  
2) [BP]를 사용 할수 없다.  
[BP]를 사용하고 싶을 때에는 [BP + 0 ]를 하면 된다  
  
간접지정할때의 주의 사항 :  
1)명령은 모두 코드 세그먼트에 있다고 본다  
2)데이터는 데이터 세그먼트 내에 있다고 가정된다.  
3)BP를 포함하는 간접지정에서는 데이터는 스택 세그먼트 내에 있다고 가정된다.  
4)프로그램의 처음에서 데이터 세그먼트의 초기설정이 필요  
  MOV   AX,DATA  
  MOV   DS,AX    ; DATA  를 데이타 세그먼트에 맟춘다.  
  
  MOV   AX,CS  
  MOV   DS,AX    ; 데이타 세그먼트를 코드세그먼트와 맞춘다.  
  
그런데 어떻게 해서라도 데이터 세그먼트 이외의 세그먼트로부터 데이터를 가져오고  
싶다는 경우 세그먼트 오버라이트 프리픽스(segment overwrite prefix)를 설정,그 명  
령에 한해서 지정한 세그먼트의 지정한 번지로 부터 데이터를 갖고 오거나,가지고 갈  
수 있읍니다.  
예)  
MOV     AX,ES:[BX]  
MOV     CS:[DI+2],CX  
세그먼트 오버라이트 프리픽스는 다음의 4가지 입니다.  
CS:,DS:,SS:,ES:  
  
  
  
  
MOV 명령으로 전송가능한 조합  
--------------------------------------------------------------------------------  
MOV     AX,CS  
MOV     DS,AX  
등 과 같은 형태로 DS   <---- CS 를 한 이유는 ,세그먼트 레지스터간의 전송이 허용  
되지 않기 때문입니다.  
또한 ,세그먼트 레지스터에는 직접 숫자를 대입 하는 것이 허용되어 있지 않습니다.  
IP,FL 레지스터는 MOV 명령에서 데이터 전송을 할수 없는 것으로 되어 있읍니다.  
CS 레지스터에의 데이터의 전송은 허용되지 않습니다.  
CS 레지스터는 읽어내기만 가능합니다.  
사칙연산 명령  
가감승제의 사칙연산을 할수가 있읍니다.  
대부분의 8비트CPU 에 서는 곱셈 나누셈을 할수가없었는데 사칙연산을 할수가 있게 된  
것은 16 비트 CPU 의 큰 특징입니다.  
  
ADD 명령의 사용법:  
ADD     AX,1234H  
현재의 AX의 내용에 숫자 1234H 가 더하여서 그 결과를 AX 레지스터에 다시 저장  
    +- 범용레지스터(8/16비트)-++- 범용 레지스터(8/16비트)-+  
ADD |  메모리      (  "     ) ||   메모리      (   "    ) |  
    +-                       -++-  숫자        (   "    )-+  
양쪽 다 메로리로 조합되는것은 허용되지않음니다.  
\*.주의 : 메모리에 수치를 더할때에는 BYTE 혹은 WORD 지정이 필요  
ADD    [BX],12H  --------> ADD   BYTE PTR [BX],12H  
이렇게 하여 바이트 혹은 워드를 지정하지 않으면 안된다.  
\*.주의 :  
AAA     EQU     32H   는 AAA 가 상수이므로 [ 숫자 ] 이다.  
BBB     DW    5678H 는 BBB가 변수이름 이므로 메모리를 나타내는 간접방식의 일종  
  
4040H+0102H 를 더하고 그결과 4142H 의 41H ,42H에 해당하는 문자를 출력한다.  
MAIN    SEGMENT  
        ASSUME  CS:MAIN  
;  
        MOV     BX,4040H  
        ADD     BX,0102H  
        MOV     DL,BH    ---+상위 바이트  
        MOV     AH,2        |  
        INT     21H      ---+  
        MOV     DL,BL    ---+하위 바이트  
        MOV     AH,2        |  
        INT     21H      ---+  
;  
        MOV     AH,4CH   ---+종류  
        INT     21H      ---+  
;  
MAIN    ENDS  
        END  
  
결과 :A>ADD1  
        AB  
      A>  
아스키 코드 41H 에 대응하는 문자 "A" 와 42H 에 대응하는 문자 "B"가 표시  
직접 숫자를 숫자로 출력하는 방법은 없는것인가 ? ---  
MS-DOS시스템호출에 숫자를 직접 출력하는 방법이 준비되어 있지않읍니다.  
  
키입력 방법  
--------------------------------------------------------------------------------  
키보드로 부터 한문자를 입력할려면 ,MS-DOS 의 펑션호출(function)의 1 번을  사용합  
니다.  AH 레지스터에 1 을 설정하고 펑션 호출을 수행하면, 키보드로부터 입력이 있  
을때 까지 기다리고 있다가 ,입력된 문자의 아스키코드를 AL register 로 돌려 보내줍  
니다.  
  
+------------------------------------------+  
|  MOV     AH,1                       &nb