제 4 장

어셈블리어 1

학습내용

* 프로그래밍 언어
* 어셈블리어 개요
* 어셈블리 프로그래밍

어셈블리어

* 컴퓨터 프로그래밍 언어의 대표적인 저급언어
  * 실행속도가 빠른 장점이 있다
* 시스템 프로그래밍에 많이 이용되는 중요한 언어
* 컴퓨터 시스템의 기본 구조와 기본 동작을 이해 하는데 도움
  * 교육적인 목적에서도 중요
* 컴퓨터 시스템에 따라 다르다.
* 8086 프로세서에서 실행되는 어셈블리어를 소개

어셈블리 언어의 필요성

* 컴퓨터 가 어떻게 작동하고 내부에서 어떻게 어떤 방식으로 계산이나 자료를 처리하는지 알고자 할 때
* 고급언어의 컴파일러가 컴파일을 효율적으로 한다고 해도 사람이 직접 어셈블리어로 작성한 것보다는 속도나 효율적인 면에서 떨어짐
  * 아주 빠른 속도를 요하는 서브 프로그램에는 어셈블리어로 작성된 프로그램을 사용하는 것이 효과적

어셈블리어의 개념

* 컴퓨터 이용하여 많이 쓰는 언어가 C언어
  * 일반 사용자는 편리하지만 컴퓨터가 이해 할 수 없다.
* 컴파일하여 기계어로 번역해서 실행
* 2진수를 사람이 이해하기 편리하도록 간단한 영어 단어를 축약하여 만듬
* 어셈블러에 의해 기계어로 변환되며 이 과정을 어셈블리라 한다.

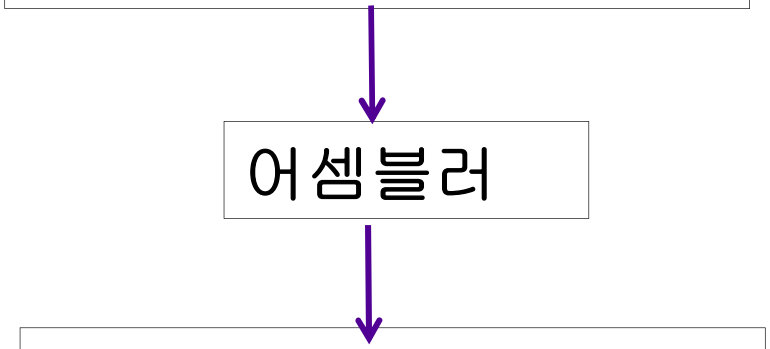
고급언어와 어셈블리어 예

```
b =1;
c= 2;
a = b+c;
```

```
mov         AX, BX
add         AX, CX
call        waitx
```

어셈블리어 의 기계어 변환 과정

```
mov        AX, BX
add        AX, CX
call       waitx
```




```
0010110011001100
1100010101000111
0001100011111100
```

원시 프로그램
어셈블리 언어로 작성

목적 프로그램
기계어로 바뀜

Ⅰ. 어셈블리어

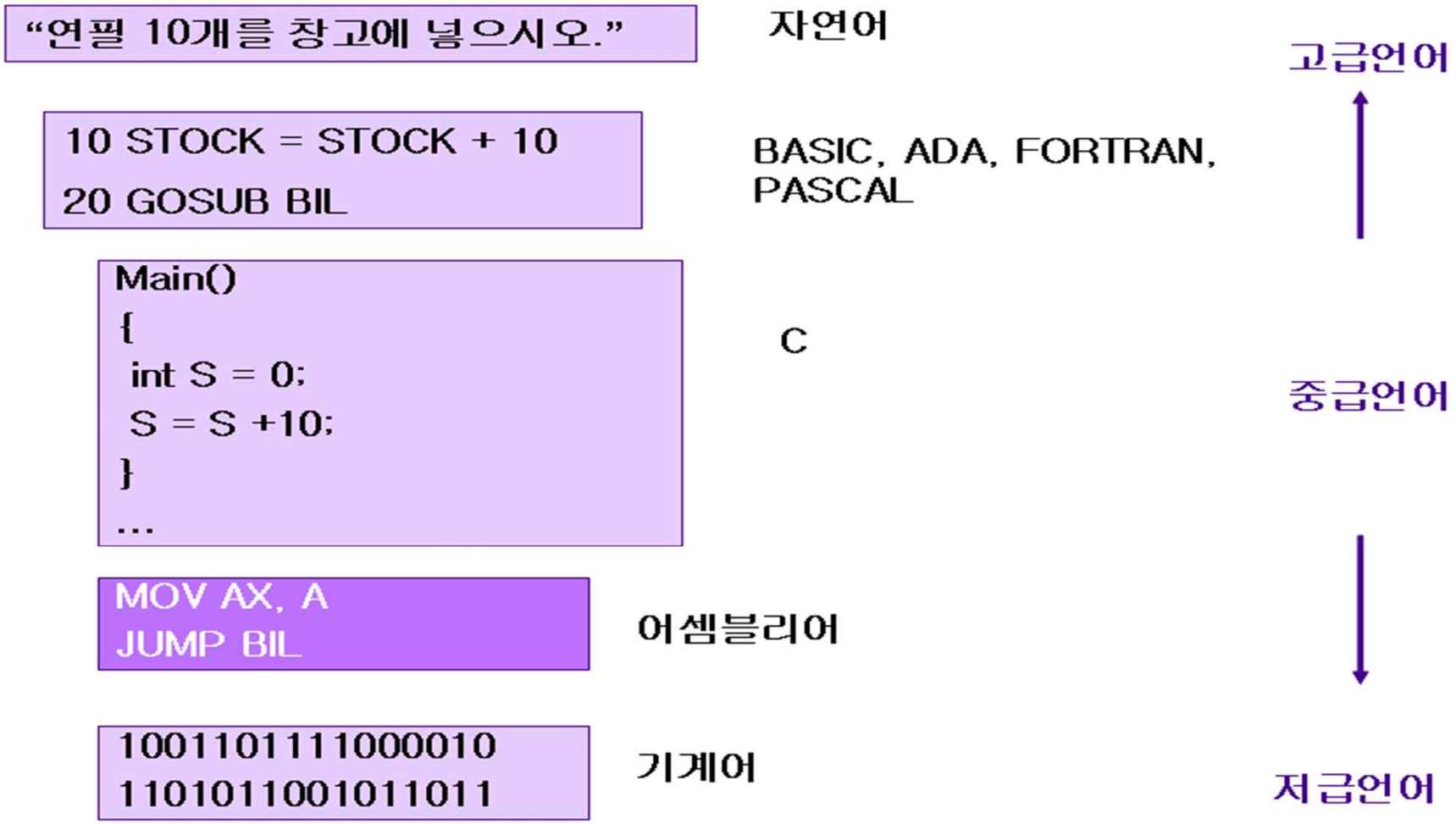
1. 프로그래밍 언어

1) 프로그래밍 언어의 계층

(1) 개요

- 프로그램 : 컴퓨터가 수행해야 할 명령어들을 순서대로 모아놓은 명령어 집합
- 프로그래밍 언어는 인간과 컴퓨터 사이에 의사 전달을 하는 수단
- 저급언어 : 기계어와 비슷한 언어
- 고급언어 : 자연어와 비슷한 언어

(2) 계층 구조


"연필 10개를 창고에 넣으시오."
자연어
고급언어
10 STOCK = STOCK + 10
20 GOSUB BIL
BASIC, ADA, FORTRAN, PASCAL
Main()
{
int S = 0;
S = S +10;
}
...
C
중급언어
MOV AX, A
JUMP BIL
어셈블리어
1001101111000010
1101011001011011
기계어
저급언어

(3) 고급언어의 분류

① 절차적 프로그래밍 언어

- 전통적인 언어
- COBOL, FORTRAN, PASCAL, C

② 객체지향 프로그래밍 언어 - 클래스, 객체, 상속의 개념

- C++, JAVA, C#, Python

③ 비주얼 프로그래밍 언어

- 윈도우 프로그래밍
- VB, VC++, Delphi

2) 고급언어

(1) 컴파일러와 인터프리터

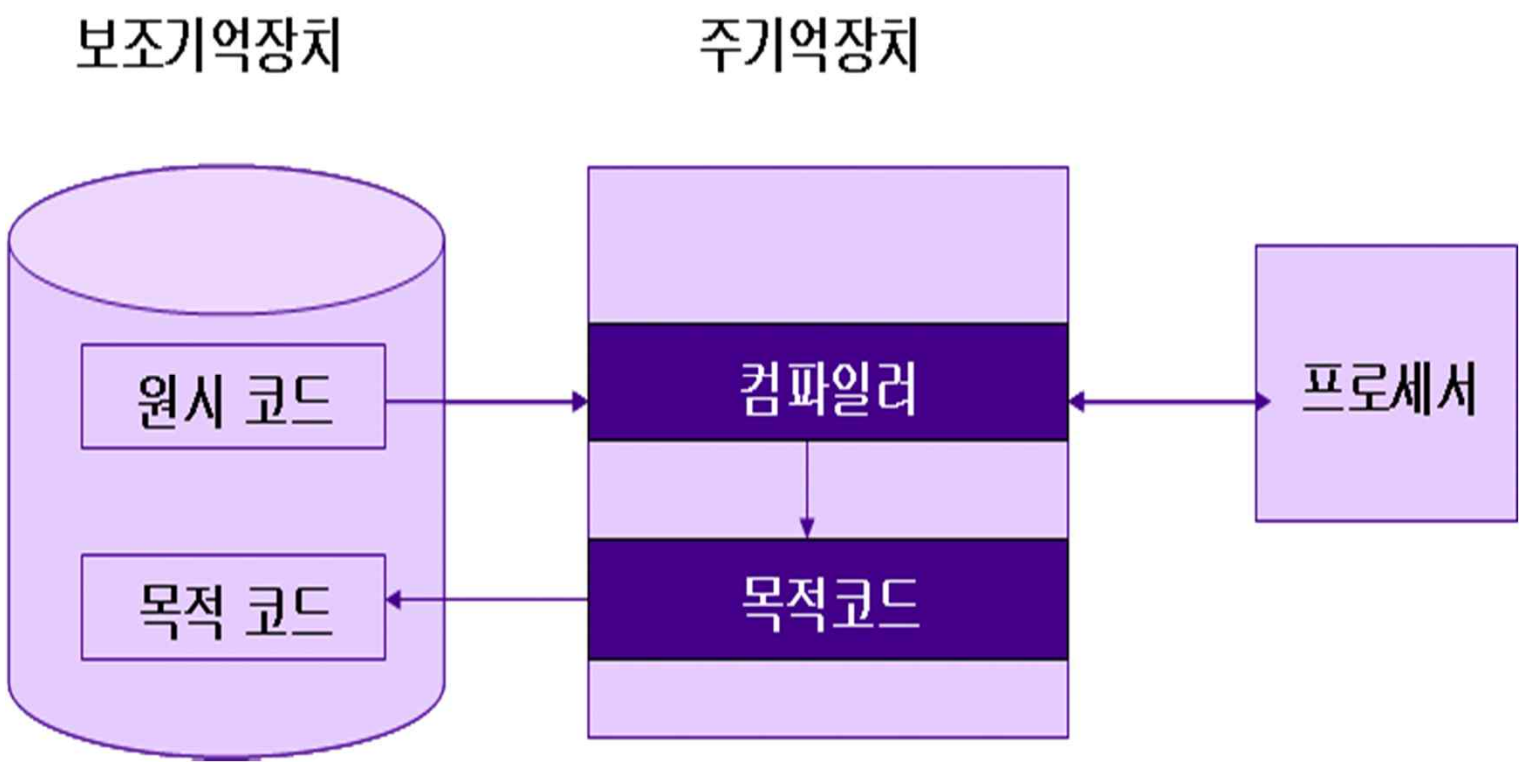
① 컴파일러

- 원시코드를 한꺼번에 목적코드로 번역하여 실행한 후 목적코드를 보조기억장치에 저장
- 컴파일시간이 많이 소요
- 실행시간이 짧아짐
- 대표언어 : Fortran, Algol, Pascal, Cobol, C, Ada

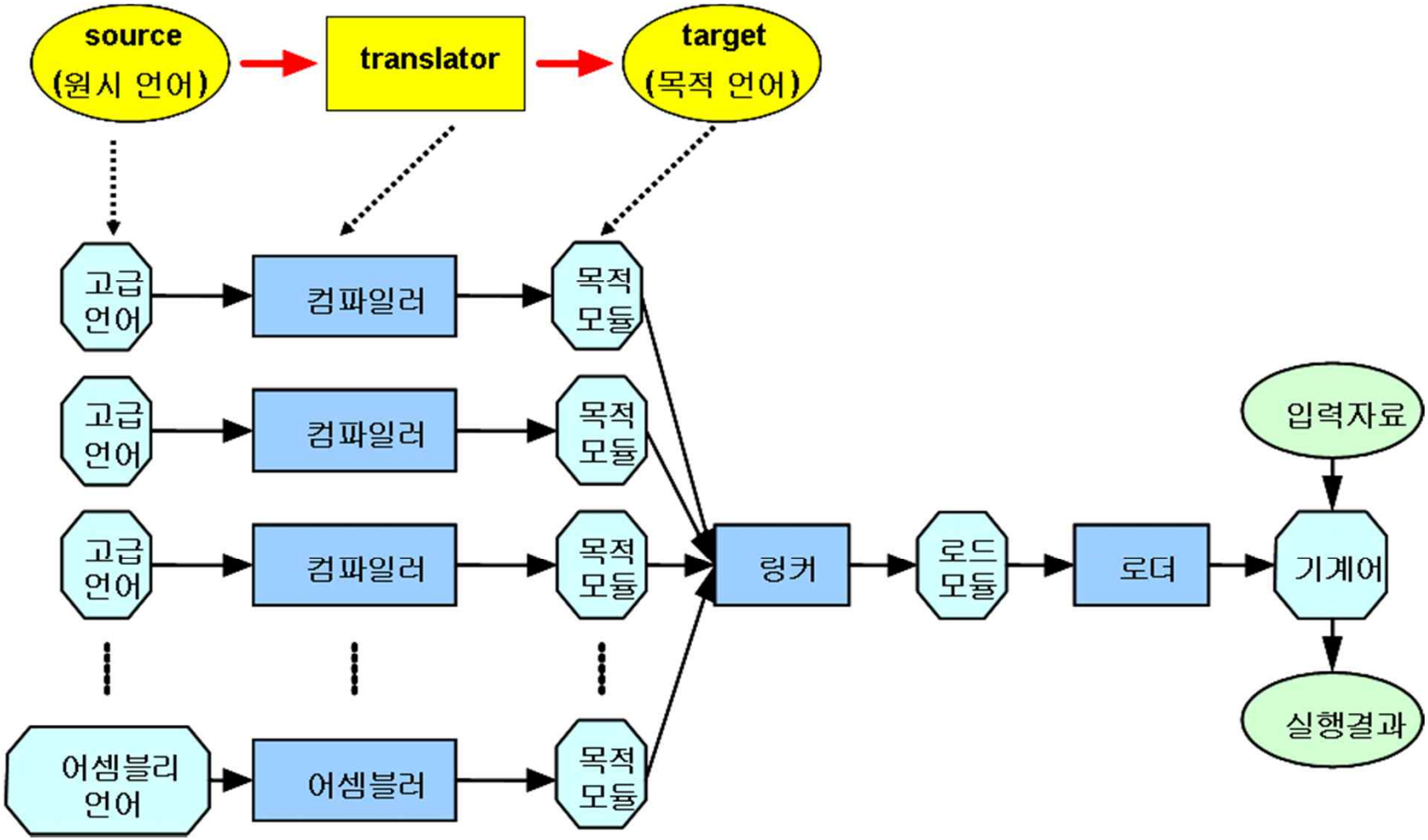
② 인터프리터

- 한 줄씩 바로 해석하여 실행하며 해석된 목적코드는 남겨지지 않고 실행할 때 마다 다시 해석
- 사용자의 유연성이 좋음
- 대표언어 : LISP, SNOBOL 4, APL, PROLOG

(2) 프로그램의 컴파일링 과정



(3) 프로그래밍 연결과 적재



3) 프로그래밍 언어의 선택

(1) 개념

- 기계어나 어셈블리어는 프로세서 별로 다르게 정의

(2) 고급언어와 저급언어

① 고급언어

- 호환성이 있고 프로그램을 이해하기 쉬움
- 컴퓨터 동작원리를 알지 못해도 배우기 쉬움

② 저급언어

- 프로그래밍 하기 힘들고 이해하기도 어려움
  상징적인 연상 기호(mnemonic instruction) 사용
- 프로그램 수행시간이 짧아짐(기계어와 1:1대응)
- 주로 시스템프로그래밍에 사용(컴파일러나 운영체제)
  UNIX 운영체제는 주로 C언어로 작성, 일부만 어셈블리
- 하드웨어 이해에 적합

어셈블리어 개요

문 장 종 류

* 어셈블리 언어 프로그램은 일련의 문장들로 구성
* 문장들의 종류

(1) 명령문(instruction)

(2) 설명문 또는 주석 문(comment)

(3) 지시어(directive)

수행문의 구성

* [Lable:] Mnemonic [Operand] [;Comment]
* 빈칸을 이용하여 구분
* 레이블은 하나의 어셈블리 명령어에 대하여 어떤 이름을 부여하는 것
  * 다른 명령어는 그 이름을 사용하여 연관된 어셈블리 명령 참조
  * 31 문자까지 가능, 콜론(:)으로 끝난다.
  * 알파벳, 숫자, 특수문자(?,,,@,$)
  * 첫 문자로 숫자는 사용하지 않음, 마침표 사용시 반드시 첫 자리에
  * 레지스터 이름은 쓰지 않는다.

수행문의 성격

* 어셈블리 지시어
  * 컴퓨터가 무엇을 하도록 지시하는 것이 아니고 어셈블러에 대한 지시 사항을 전달
  * 기계어로 번역되지 않는다
* 어셈블리 명령어
  * 컴퓨터로 하여금 무엇을 수행하라고 직접 명령
  * 기계어로 번역
  * 연산코드와 연산 항으로 이루어짐

수행문의 구성

* 연산코드 : 실행 내용을 나타냄

  어셈블리 언어를 배우는 것

* 연산 항 : 명령어에 따라 다름
  * 명령어가 작용하는 레지스터나 기억 장소의 위치
  * 두 개 일 때, 앞의 것은 목적지 연산 항 뒤의 것은 출발지 연산 항

어셈블리어 명령어 형식

1) 명령어 형식(p104)

(1) 어셈블리 지시어

- 기계어로 번역되지 않고 단지 어셈블러에게 특정한 작업을 지시(실행x)

(2) 어셈블리 명령어

- 명령어는 실제로 어떤 작업을 수행
- 연산코드(operation code)와 연산 항(operand)으로 이루어짐
- 연산 항은 없을 수도 있고, 1개 - 3개까지 존재

① 0개의 연산항 : <op-code>

예) CLC

② 1개의 연산항 : <op-code> <operand>

예) DEC CX

③ 2개의 연산항 : <op-code> <operand 1> <operand 2>

예) MOV AX, BX

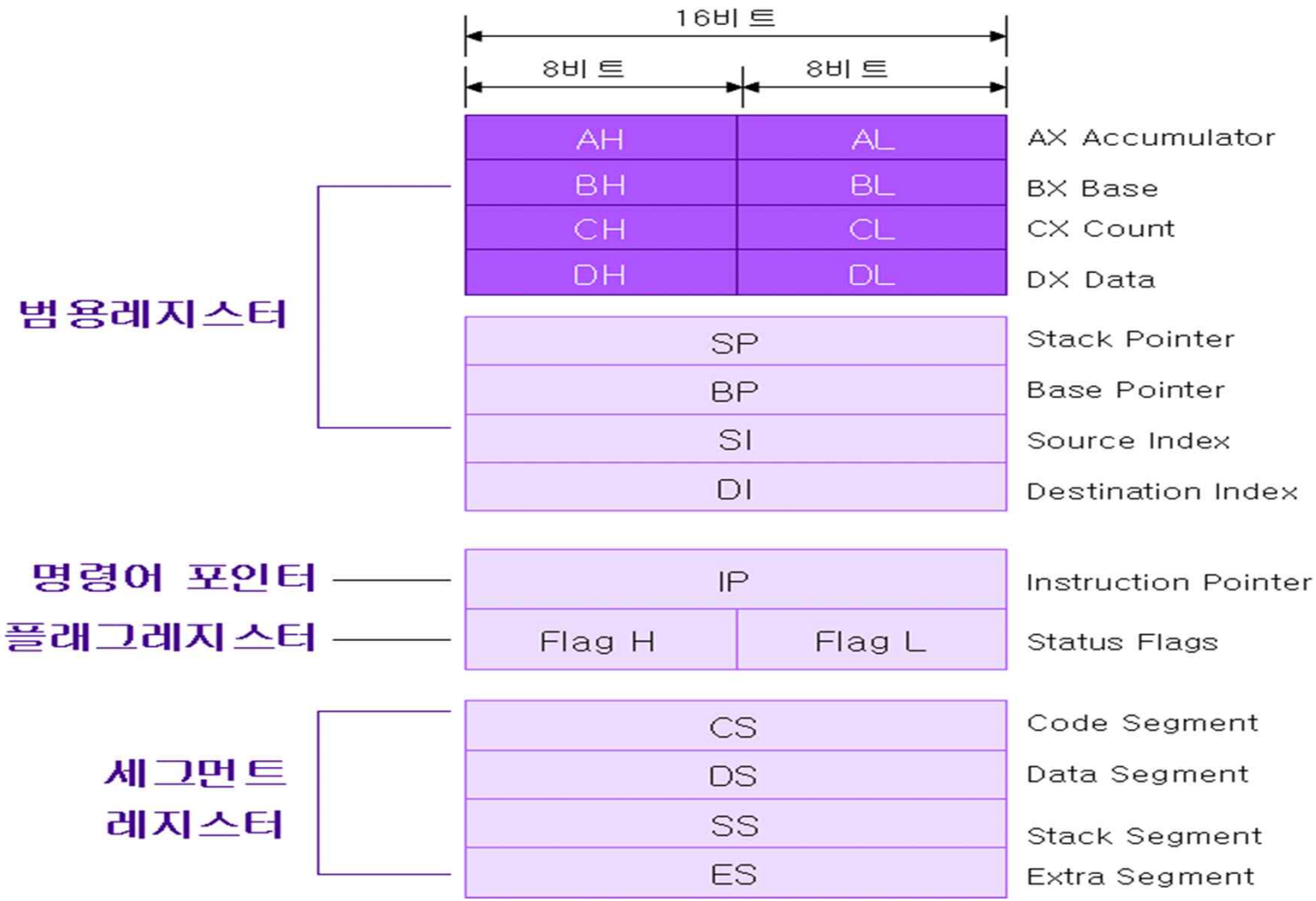
<op-code> : JMP, ADD, MOV 와 같이 명령어가 어떤 일을 해야 하는 지 알려주는 연상기호

<operand> : 명령어가 작용하는 레지스터나 기억장소의 주소 나타냄

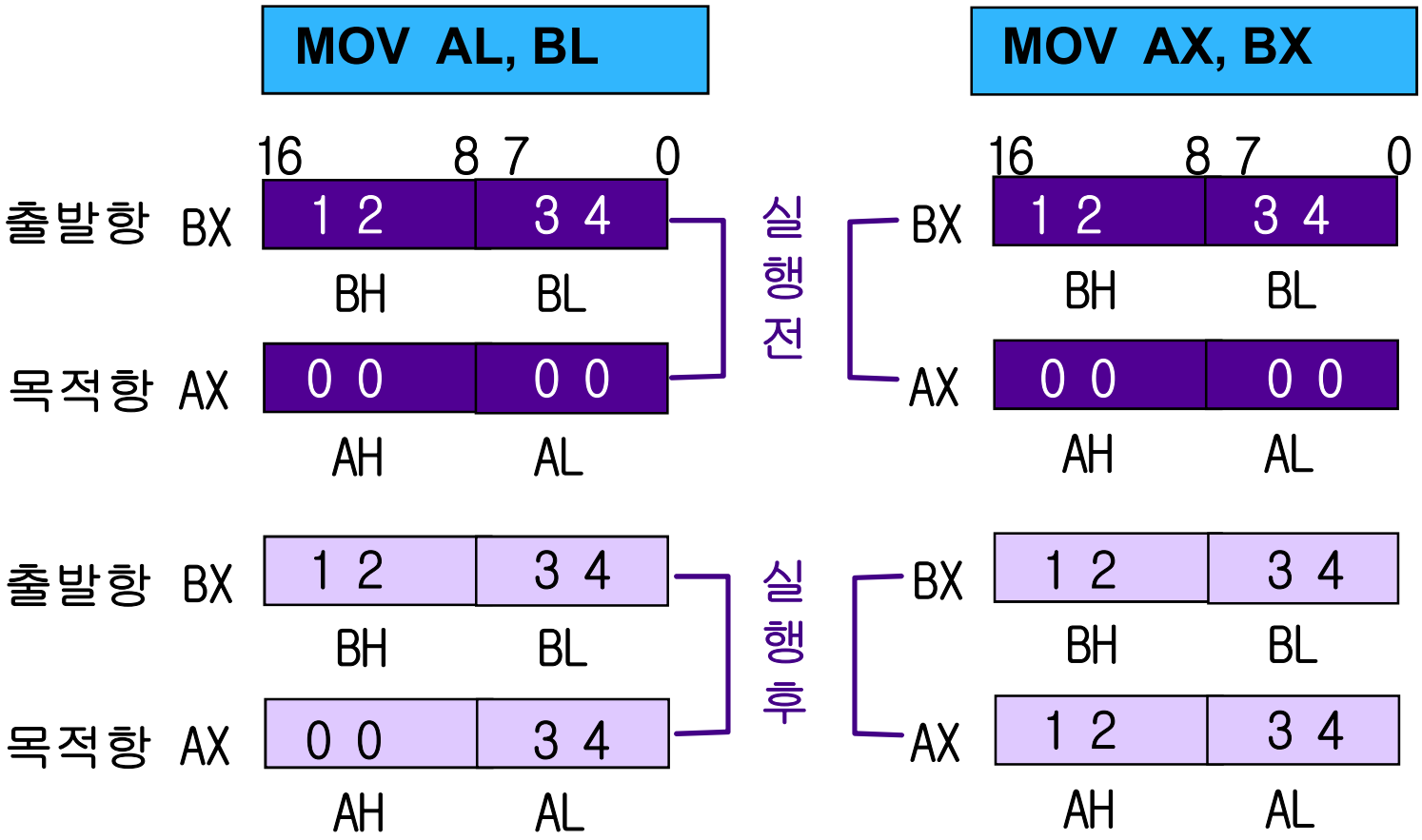
연산 항이 두 개 일 때는 앞의 것을 목적지 항, 두 번째 것을 출발지 연산 항

연산결과는 목적 항에 기록, 연산 후 출발 항은 변하지 않고 목적항만 변함

※ 레지스터 구성도


16비트
8비트
8비트
AH
AL
BH
BL
CH
CL
DH
DL
AX Accumulator
BX Base
CX Count
DX Data
범용레지스터
SP
BP
SI
DI
Stack Pointer
Base Pointer
Source Index
Destination Index
명령어 포인터
IP
Instruction Pointer
플래그레지스터
Flag H
Flag L
Status Flags
세그먼트
레지스터
CS
DS
SS
ES
Code Segment
Data Segment
Stack Segment
Extra Segment

(3) 2개의 연산항 명령어



a) 바이트 연산

b) 워드 연산

화면에 A문자를 출력하는 프로그램

```
1   MAIN  SEGMENT
2         ASSUME  CS : MAIN
3   ;
4         MOV  DL, 'A'
5         MOV  AH, 2
6         INT  21H
7   ;
8         MOV  AH, 4CH
9         INT  21H
10  ;
11        MAIN  ENDS
12        END
```

화면에 A문자를 출력하는 프로그램

```
1  MAIN SEGMENT
2        ASSUME CS:MAIN
3 ;
4        MOV DL, 'A'
5        MOV AH, 2
6        INT 21H
7 ;
8        MOV AH, 4CH
9        INT 21H
10;
11       MAIN ENDS
12       END
```

① MOV <Operand 1>, <Operand 2>
- 연산 항에는 레지스터, 메모리 주소, 상수 등 올 수 있음

② 6번 문장의 INT 21H 명령
- 21H번의 인터럽트를 호출하는 명령
  강제적으로 CPU가 하던 일을 중단시키고 MS-Dos에 의한 입출력 등의 시스템 함수의 호출에 사용
  AH레지스테에 들어 있는 값에 따라 다른 기능 수행

AH가 2 일때 :DL레지스터에 저장되어있는 ASCII코드에 해당하는 문자가 화면상에 표시

③ 8, 9번 문장
- 4CH를 레지스터 AH에 저장하고
  INT 21H명령으로
  인터럽트 번호 21H를 호출하여
  4CH에 해당되는 기능 수행

AH가 4CH 일때 :실행중의 프로그램을 끝내고 DOS상으로 되돌아가라는 명령

※ DOS 함수 호출
  2 : 콘솔로 한 문자를 출력
  4C : 프로그램 종료

(4) 어셈블리어 프로그램 형태

```
MAIN   SEGMENT
                ASSUME   CS:MAIN

                  자신이 만든 프로그램 블록

MAIN   ENDS
                END
```

① MAIN SEGMENT

- 세그먼트이름을 'MAIN'으로 선언 (어셈블리 지시어)

② ASSUME CS:MAIN

- 어셈블러가 세그먼트를 참조하는 명령을 생성할 때
- 예)'MAIN'이라는 이름의 세그먼트를 참조하라

③ 자신이 직접 코딩한 프로그램

④ MAIN ENDS : 세그먼트의 끝을 나타냄

⑤ END : 프로그램의 끝을 나타냄

MOV 명령어

MOV < operand1 > , < operand2 >

MOV 명령과 문자 출력

데이터의 입출력과 전송명령에 대해서 해설 합니다.

* .레지스터에 수치를 대입한다.

* .레지스터와 레지스터 사이에서 데이터를 전송한다.

* .레지스터와 메모리 사이에서 데이터를 전송한다.

- MOV 연산항에는 레지스터, 메모리 주소, 상수 등이 올 수 있다.
- 출발항 과 목적항의 결과에 따라 몇 가지 조합만 가능

MOV명령 연산항의 가능한 조합

MOV < operand1 > , < operand2 >

operand1 \ operand2	값	직 접		간접
		범용r.	세그먼트r.	
범용 r.	○	○	○	○
세그먼트 r. (CS는 제외)	×	○	×	○
간접:메모리주소	○	○	○	×

주석문(설명문)

★ 설명문: 한 문장 전체가 될 수도 있고, 한 문장의 일부분일 수도 있음
- 어셈블러는 어셈블 할 때 이 설명문을 무시하고 지나간다.
- 영문, 한글 가능 (프로그램에 설명을 삽입하기 위해 사용됨)

* ★설명문(주석)
* (1) 세미콜론(;)을 문장의 맨 앞에서 두어서 사용
* (2) 명령어 맨 마지막에 세미콜론(;)을 사용하는 방법

주석문(설명문)

★설명문의 형식:

* 한 문장의 공백이 아닌 첫째 문자가 세미콜론(;)인 경우
  * ; This line is a comment
* 각 명령문이나 지시어의 뒤에도 설명문이 올 수 있다.
  * 예) MOV AX, 99 ; Here is a comment
* 어셈블러는 프로그램의 한 줄을 처리할 때 세미콜론이 나타나면 그 뒤의 문자열을 무시한다는 것이다.
* 예외) 세미콜론이 따옴표(double quote)나 작은 따옴표(single quote)에 둘러싸여 있는 경우이다.
  * "hello; goodbye"
  * 'hello; goodbye'
* 빈 줄(blank line)도 설명문임

데이터 정의 지시어

데이터 정의

(1) 상수 정의

① 숫자상수 예

- 이진수 : 01010111b
- 8진수 : 317o 혹 514q
- 10진수 : 792 혹 792d
- 16진수 : 4F7h, 0adH
- 부동소수점 : 427E-3

② 문자/스트링 상수의 예

- 'Y', 'yang', 'soo'

③ 상수 정의시 EQU 나 = 사용

- EQU는 재정의 불가능, = 재정의 가능

<선언>

레이블 EQU 상수(식)

레이블 = 상수(식)

<예>

SUM EQU 2FFH

XT = 'Z'

SUM EQU 3FAH : 오류

XT = 'Y'

SUM = $2FF_{(16)}$

상수

* 정수
  * 표시는 대문자 또는 소문자(b; o,q; d; h)
  * 16진법 첫 숫자는 반드시 아라비아 숫자
    * abcH → 0abcH
  * 16진법에서 A,B,C,D,E,F는 대/소문자
  * 아무런 표시가 없으면 10진수
* 문자 상수
  * 'A' 'AB' 'Can"t' "Can't"
* 실수 : 12.34, 1.234E1
  * DD, DQ, DT에 의해서만 정의 가능
  * 실수 자체가 명령어에서 직접 사용되지 않음

데이터 정의 지시어

(2) 변수 정의

<형식>

```
[레이블]   DB 식[,반복]    ; 바이트(8)
           DW 식[,반복]    ; 워드(16)
           DD 식[,반복]    ; 2워드(32)
           DQ 식[,반복]    ; 4워드(64)
           DT 식[,반복]    ; 10바이트(80)
```

<예>

```
AAA   DB   15              AAA = 15
BBB   DB   'a', 0BFH       BBB = 'a' 와 0BF(16) 가 차례로 저장
      DW   34EFH           레이블 없이 저장
CCC   DD   ?               초기값이 없음
DDD   DQ   12AB34CD56EF78H 4워드 정의
EEE   DB   2 DUP(3)        EEE  DB 3, 3
```

어셈블리 지시어

* 프로그램 출력 지시어
* 프로세서 관련 지시어
* 데이터 관련 지시어

프로그램 출력 지시어

* 프로그램 출력 지시어

  PAGE [lines] [,columns] ; 프린트 될 페이지 결정

  TITLE text(임의의 문장)

  SUBTTL text(임의의 문장)

* lines : 10~255 , default : 57
* columns : 60~132, default : 80
* TITLE의 text 내용은 각 페이지 둘째 줄 왼쪽
* SUBTTL의 text 내용은 셋째 줄 오른쪽

데이터 관련 지시어

* 데이터 관련 지시어
  * 데이터 정의(DB, DW, DD, DQ,DT)
  * 심볼 정의(EQU, =)
  * 외부 프로그램 참조(PUBLIC, EXTRN, INCLUDE)
  * 영역 정의(SEGMENT, ASSUME, PROC)
  * 어셈블리 제어(END, EVEN, ORG)

데이터 정의 지시어

* DB(Byte) : 1 바이트 이상의 메모리 할당
* DW(Word) : 2 바이트 메모리 할당
* DD(Double Word) : 4 바이트 메모리 할당
* DQ(Quad Word) : 8 바이트 메모리 할당
* DT(Ten Bytes) : 10 바이트 메모리 할당

[name] DB expression[,...]

데이터 정의 지시어

* Max DB 255
* wmax DW 65535
* table DB 0,1,2
* table DB 0
  DB 1
  DB 2
* table DW 3, 3, 3, 3, 8, 7, 7, 7
  table DW 4 DUP(3), 8, 3 DUP(7)
* temp DB ?
* temp1 DW 12 DUP(?)
* 문자는 DB 사용 : msg DB 'abcabc'
  msg DB 2 DUP('abc')

데이터 정의 지시어

* AA DW 10 DUP(?)
* BB DB 5 DUP(14)
* CC DB 3 DUP(4 DUP(8))

심볼 정의 지시어

* EQU : 프로그램 내에서 변경 불가

  sum EQU 2FFh

  sum EQU 3FAh; 재정의 오류

* = : 프로그램 내에서 변경 가능

외부 프로그램 참조 지시어

* 여러 프로그램을 연결 편집하여 하나의 수행 가능한 프로그램을 만들 때 각각의 프로그램 간에 서로 정보를 주고 받아야 한다.
* PUBLIC symbol[,...]
  * 외부 사용자가 이 프로그램을 사용해도 좋다
* EXTRN name : type[, ...]
  * 다른 곳에서 정의된 것을 사용하겠다
  * type : byte, word, dword, ABS(EQU, =)

영역 정의 지시어

- SEGMENT
  - 프로그램의 영역을 구분하는데 사용
  - 세그먼트란 한 세그먼트 레지스터 내의 주소를 참조 함으로서 찾아갈 수 있는 모든 명령문과 데이터의 집합
  - s_name SEGMENT

    …

    s_name ENDS
- ENDS
- ASSUME
  - ASSUME s_reg : s_name

어셈블리 제어 지시어

* END [entry_point name]
  * 원시 프로그램의 끝을 정의
* EVEN
  * 메모리 주소를 짝수에 위치시킴

연산자

- 산술 연산자(+, -, *, /, MOD, SHL, SHR)
  - / : 몫, MOD: 나머지
  - abc EQU 00110010B SHL 1
  - def EQU abc SHR 1
- 논리 연산자(AND, OR, XOR, NOT) : 비트 단위
- 관계연산자(EQ, NE, LT, GT, LE, GE)

응용1. 화면에 문자 “AB” 를 출력하는 프로그램