3. 어셈블러 프로그램 형식

3.1 8086 어셈블러
3.2 프로그램 형식
3.3 지시자
3.4 수식
3.5 상수
3.6 프로그램 구조

3.1 8086 어셈블러

- MASM(Microsoft Macro Assembler)
  - 인텔 문장 구문법 사용
  - 피연산자 순서는 목적지, 출발지 순으로 작성
  - add eax, ebx ; eax = eax + ebx
  - 즉치값은 데이터를 그대로 표시
  - Mov eax, 1 ; 10진수 1을 eax에 저장
  - Mov eax, 1001B ; 2진수 1001을 eax에 저장
  - Mov eax, 175o ; 8진수 175를 eax에 저장
  - Mov eax, 1fedh ; 16진수 1fed 를 eax에 저장
  - 레지스터에는 아무런 표시를 하지 않는다.
  - Mov al, “A” ; 영문자 A를 al에 저장
  - B = 1바이트, W = 2바이트, D = 4바이트,
  - Q = 8바이트, T = 10바이트

3.1 8086 어셈블러

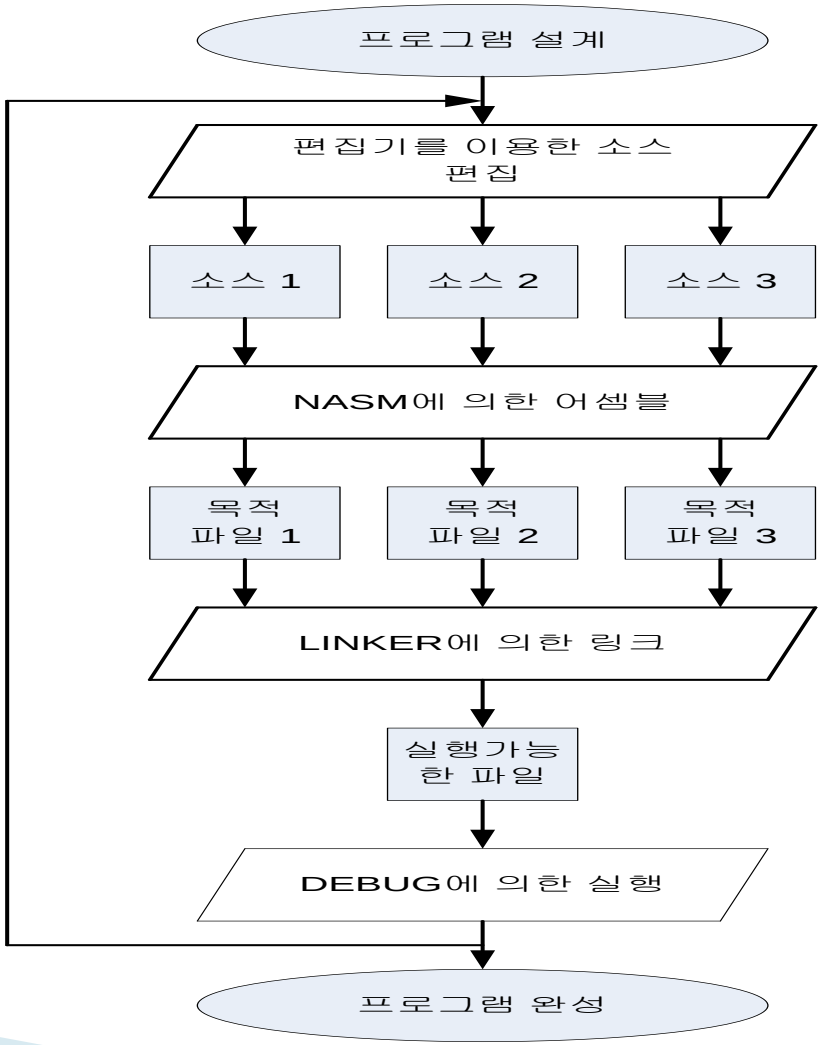
- GAS(GUN Assembler)
  - AT&T 문장 구문법 사용
  - 피연산자 순서는 출발지, 목적지 순이다.
  - Add %eax, %ebx ; ebx = ebx + eax
  - 즉치값에는 $를 레지스터에는 %를 붙인다.
  - Mov $1, %eax ; 1을 eax에 저장
  - B = 1바이트, W = 4바이트, L = 8바이트

3.1 8086 어셈블러

- NASM(Netwide Assembler)
  - 인텔 구문 표기법 사용.
  - MASM과 매우 유사.
  - MASM은 MS-DOS에서 동작, Linux에서는 동작하지 않는다.
  - GAS는 Window에서 동작하지 않는다.
  - NASM은 Window 및 Linux 모두에서 동작 한다.

3.1 8086 어셈블러

- 어셈블링, 링킹, 실행



3.2 프로그램 형식

- 프로그램 주석(Comment)
  - 주석은 프로그램의 명확성을 향상.
  - 레지스터 사용에 따른 프로그램 목적의 불 명확성.
  - 어셈블리어의 주석은 세미콜론(;)으로 같은 줄 오른쪽은 전부 주석처리 된다.
  - 주석은 소스 프로그램에서만 존재하고 기계어로 번역 되지 않는다.
  - 모든 명령어에 주석처리 하는 습관을 기른다.

3.2 프로그램 형식

- 예약어(Reserved word)
  - 어셈블리에서 고유한 목적을 위해 예약되어 있으므로 특정 조건에서만 사용 가능.
  - Mov, add와 같은 명령어.
  - 어셈블러에게 정보를 제공하는 equ와 같은 지시어.
  - 식에서 사용하는 +, -와 같은 연산자.
  - 어셈블러 과정 중 프로그램에 정보를 전해주는 %define 등과 같은 미리 정의된 기호.
  - 예약어는 기능에 맞지 않게 잘못 사용하면 오류 메시지 발생.

3.2 프로그램 형식

- 식별자(Identifier)
  - 식별자는 프로그램에서 참조할 항목을 표현하기 위해 사용하는 이름으로 식별자는 이름과 라벨로 분류.
  - 이름(name)은 데이터 항목의 주소를 참조.
  - cnt db 0 ; 0을 메모리 주소 cnt에 저정
  - 라벨(label)은 명령어, 프로시저, 세그먼트 주소를 참조.
  - _asm_main:
  - init: add al, 1 ; init는 덧셈 명령어 주소를 참조.
  - 식별자 만드는 규칙은 영문자 A~Z, a~z, 숫자 0~9(첫 문자로 사용 불가), 특수문자(?, _, $, @, .,) (첫 문자로 사용 불가)
  - 레지스터는 식별자로 사용불가.

3.2 프로그램 형식

- 문장(Statement)
  - 어셈블리어는 문장의 집합으로 구성, 문장에는 2가지 유형.
  - 어셈블러가 목적코드로 번역하는 명령어(mov, add 등)
  - 어셈블러가 특정 행동을 수행하도록 알려주는 지시어.

[식별자]	연산	[피연산자들]	[;주석]

```
지시어   CNT     DB      1          ; 이름, 연산, 피연산자
명령어   L30:    MOV     AX, 0      ; 라벨, 연산, 두 개의 피연산자
         ret                        ; 프시저 복귀
         inc     bx                 ; bx 레지스터 값을 1증가
         add     cx, 10             ; cx = cx + 10
```

3.3 지시자

- 지시자(Directive)
  - 지시어는 어셈블러를 위해 정보를 제공.
  - 지시어는 기계어 코드로 변환되지 않는다.
  - 상수 정의.
  - 메모리 정의.
  - 메모리 세그먼트 정의.
  - 조건적으로 소스 컴파일.
  - 다른 파일 포함.
  - NASM의 전처리는 C와 다르게 #이 아닌 %를 사용.

3.3 지시자

- equ 지시자
  - 심볼(symbol)을 정의 할 때 사용.
  - 심볼은 어셈블리 프로그래밍에서 상수를 뜻한다.
  - symbol equ value
  - 한번 정의된 심볼은 절대로 재정의 될 수 없다.
- %define 지시어
  - C언어와 유사, 상수 매크로 정의에 사용.
  - %define SIZE 100
  - mov eax, SIZE
  - 매크로는 심볼들보다 융통성이 있어 재정의 될 수 있다.
  - 단순히 상수정의에도 사용.

3.3 지시자

- 데이터 지시자
  - 데이터 세그먼트에서 메모리 상의 공간을 정의.
  - 데이터를 위한 공간만 정의하는 방법과 데이터 공간을 지정하고 초기값을 같이 정의하는 방법이 있다.
  - RESx 지시어는 x에 저장될 데이터의 크기를 나타냄.
  - Dx 지시어는 초기값을 같이 정의, x에 저장될 데이터 크기를 나타냄.

데이터 크기	문자
바이트	B
워드	W
2중 워드	D
4중 워드	Q
10바이트	T

3.3 지시자

- L1 : db 0 ; L1 라벨에 0으로 바이트 초기화
- L2 : dw 1000 ; L2 라벨에 1000으로 워드 초기화
- L3 : db 110101b;L3 라벨에 110101로 바이트 초기화
- L4 : db 12h ; L4 라벨에 16진수 12로 바이트 초기화
- L5 : db 17o ; L5 라벨에 8진수 17로 바이트 초기화
- L6 : dd 1a92h;L6 라벨에 16진수 1a92로 2중워드 초기화
- L7 : resb 1; L7 라벨에 초기화 되지 않은 1바이트 정의
- L8 : db “A” ; L8 라벨에 아스키코드 A(65) 바이트 초기화
- L9 : db 0, 1, 2, 3; L9에서 순차적으로 4바이트 정의
- L10 : db “W”, “O”, “R”, “D” ; L10에 “WORD” 정의
- L11 : db ‘WORD’, 0 ; L10과 같은 정의

3.3 지시자

- TIMES 지시자
  - 피연산자를 지정한 횟수 만큼 반복.
  - 초기화를 하지 않을 때는 RESx 지시어를 사용.
  - L12 : times 100 db 0; 100개의 0으로 바이트 초기화.
  - L13 : resw 100 ; 100개의 워드 공간 정의.
  - 라벨은 데이터의 주소 또는 옵셋으로 사용.
  - 대괄호([]) 속에 라벨을 사용하면 그 주소의 값을 의미.

3.3 지시자

- mov al, [L1] ;L1주소의 바이트 값을 al 레지스터로 저장.
- mov eax, L1 ; eax에 L1의 주소를 저장.
- mov [L1], ah ; ah 바이트 값을 L1 번지에 저장.
- mov eax, [L6] ; L6주소의 2중 워드를 eax에 저장.
- add eax, [L6] ;L6의 2중 워드를 eax와 더한 후 eax에 저장.
- add [L6], eax ; L6의 2중 워드를 eax와 더한 후 L6 주소에 저장.

3.3 지시자

- 연산자 크기 지시자
  - mov al, [L6] ; 에러 메시지 출력
  - mov [L6], 1 ; 에러 메시지 출력
  - 어셈블러는 라벨이 참조하는 데이터 형식과 레지스터 크기를 맞게 조정하지 않는다.
  - mov dword[L6], 1
  - 1을 메모리 L6이 가리키는 곳에 2중 워드 크기로 저장.
  - 데이터 크기를 규정하는 지시어는 다음과 같다.
  - BYTE = 1바이트, WORD = 2바이트,
  - DWORD = 4바이트, QWORD = 8바이트,
  - TWORD = 10바이트(부동소수점 프로세서가 사용).

3.3 지시자

- segment 지시자
  - 메모리는 세그먼트 단위로 프로그램 코드, 데이터, 스택으로 나누어진다.
  - 적용 방법은 다음과 같다.
  - segment stack ; 스택 영역
  - segment data ; 초기화된 데이터 영역 또는 데이터 영역
  - segment bss ; 초기화 되지 않은 데이터 영역
  - segment code ; 프로그램 코드 영역

3.3 지시자

- EXTERN 지시자
  - EXTERN 지시어는 C언어의 extern과 같다.
  - 다른 모듈에서 이미 GLOBAL로 선언된 심볼이나 라벨을 현재 모듈에서 EXTERN 지시어를 사용하여 외부에서 선언된 심볼이나 라벨이라는 것을 알려 준다.
  - extren _printf
  - extren _printf, _scanf
  - 심볼이나 라벨은 코마(,)로 구분하여 여러 개 연속하여 사용 가능.

3.3 지시자

- GLOBAL 지시자
  - GLOBAL 지시어를 사용해 심볼이나 라벨을 전역변수로 사용.
  - GLOBAL로 선언된 심볼이나 라벨은 프로그램 다른 모듈에서 사용 가능.
  - 당연히 GLOBAL로 먼저 선언 후 사용.
  - global _main
  - _main:

3.3 지시자

- INCBIN 지시자
  - INCBIN 지시어는 외부 2진 파일을 프로그램에 포함.
  - incbin “file.txt” ; file.txt를 전체 포함.
  - incbin “file.txt”, 1024 ; file.txt 첫 1024바이트 skip 후 나머지 전체.
  - incbin “file.txt”, 1024, 512 ; file.txt 첫 1024바이트 skip 후 512바이트 포함.

3.4 수식

- NASM의 수식은 C와 매우 유사.
  - | ; OR 연산자.
  - ^ ; XOR 연산지.
  - & ; AND 연산자.
  - <<, >> ; 왼쪽, 오른쪽 쉬프트 연산자.
  - +, - ; 덧셈, 뺄셈 연산자.
  - *, /, //, %, %% ; 곱셈, unsigned 나눗셈, signed 나눗셈, unsigned 나머지, signed 나머지 연산자.
  - 단항 연산자 +, -, ~, ! ; 양수, 음수, 1의 보수, 논리 부정 연산자.

3.5 상수(10진법, 진법 표현)

- 수치 상수.
  - 접미사 H, X는 16진수, Q, O는 8진수, B는 2진수.
  - 접두사 0x, 0H, $는 16진수, 0O, 0Q는 8진수, 0B는 2진수.

```
mov     ax, 200          ; 십진수 200
mov     ax, 0200         ; 십진수 200
mov     ax, 0200d        ; d는 십진수를 의미
mov     ax, 0d200        ; 0d 역시 십진수
mov     ax, 0c8h         ; h는 16진수 c8, 16진수는 숫자로 시작 함
mov     ax, $0c8         ; $는 16진수, $다음에 반드시 0과 함께 사용
mov     ax, 0xc8         ; 0x는 16진수
mov     ax, 0hc8         ; 0h는 16진수
mov     ax, 310q         ; q는 8진수
mov     ax, 310o         ; o는 8진수
mov     ax, 0o310        ; 0o는 8진수
mov     ax, 0q310        ; 0q는 8진수
mov     ax, 11001000b    ; b는 2진수
mov     ax, 1100_1000b; _는 긴 2진수의 구분자(break up)
mov     ax, 0b110_1000; 0b는 2진수
```

3.5 상수(10진법, 진법 표현)

- 문자 상수.
  - 역 슬래시(\)를 사용.
  - \' ; 따움표(')
  - \" ; 쌍 따움표(")
  - \\ ; 역 슬래시(\)
  - \? ; 물음표(?)
  - \a ; BEL (ASCII 7)
  - \b ; BS (ASCII 8)
  - \t ; TAB (ASCII 9)
  - \n ; LF (ASCII 10)
  - \v ; VT (ASCII 11)
  - \f ; FF (ASCII 12)
  - \r ; CR (ASCII 13)
  - \e ; ESC (ASCII 27)

3.5 상수(10진법, 진법 표현)

- 문자열 상수.
  - ‘와 ‘, “와 “사이의 모든 문자열을 같은 문자열 상수로 취급.
  - db 'hello', 0 ; hello 정의, 문자열의 끝은 ‘0’이다.
  - db 'h’, ‘e’, ‘l’, ‘l’, ‘o', 0 ; hello 정의, 문자열의 끝은 ‘0’ 이다.
  - db "hello", 0 ; 위의 문장과 동일.

3.5 상수(10진법, 진법 표현)

- 부동소수점 상수.
  - dd 0x1p+2 ; $1.0 \times 2^2 = 4.0$
  - dq 0x1p+32 ; $1.0 \times 2^{32} = 4294967296.0$
  - dq 1.e10 ; 10000000000.0
  - dq 1.e+10 ; 위 예와 같다.
  - dq 1.e-10 ; 0.0000000001

3.6 프로그램 구조

- NASM 골격 프로그램.

```
segment stack stack; stack 영역
resw 256
tos:
segment data  ; 초기화 된 데이터를 정의하는 부분
          ; 초기화 되지 않은 데이터를 정의하는 부분
segment code    ; 소스 코드를 작성하는 부분
..start:        ; 프로그램 진입 점
mov ax, data    ; 데이터 세그먼트 주소를 ax에 저장
mov ds, ax      ; 데이터 세그먼트 주소 셋팅
mov es, ax      ; es에 데이터 세그먼트 주소 셋팅
cld             ; 스트링 명령어는 오름차순으로 사용
cli       ; stack을 정의하기 위하여 인터럽트 불능
mov ax, stack ; SS 지정
mov ss, ax    ;
mov sp, tos   ; SP 지정
sti           ; 인터럽트 가능
      ; 각자 필요한 목적에 맞는 프로그램 코드 작성
mov ax, 4c00h ; 프로그램 종료. DOS로 제어를 넘김
int 21h       ; MS-DOS 호출
```

3.6 프로그램 구조

- NASM 첫 예제 프로그램.
  - 키보드로부터 한 문자를 입력 받아 화면에 출력 후 줄바꿈을 한 다음 입력 받은 문자를 다시 화면에 출력하는 프로그램.

```
        segment  stack      stack                          ; stack 영역
        resw     256
tos:

        segment  data                                       ; 데이터 세그먼트 시작점
prompt1         db       " input1 : $“                      ; 입력화면 설계
prompt2         db       0dh, 0ah, " output : $"            ; 줄바꿈과 출력화면 설계

        segment           code                              ; 코드 세그먼트 시작점

..start:                                                    ; 프로그램 진입점
        mov      ax, data                                   ; 프로그램 골격
        mov      ds, ax                                     ; 프로그램 골격
        mov      es, ax                                     ; 프로그램 골격
        cld                                                 ; 프로그램 골격
        cli                                                 ; 프로그램 골격
        mov      ax, stack                                  ; 프로그램 골격
        mov      ss, ax                                     ; 프로그램 골격
        mov      sp, tos                                    ; 프로그램 골격
        sti                                                 ; 프로그램 골격
```

3.6 프로그램 구조

```
mov     ah, 09h             ; 설계된 입력화면 출력을 위한 MS-DOS 기능
mov     dx, prompt1         ; 문자열의 화면 출력을 위한 주소 셋팅
int     21h                 ; MS-DOS 호출

mov     ah, 01h             ; 한 문자를 키보드로부터 입력받는 기능
int     21h                 ; MS-DOS  호출

mov     bl,     al          ; 키보드로부터 입력 받은 문자를 bl에 저장
mov     ah, 09h
mov     dx, prompt2
int     21h
mov     ah, 02h             ; 한 문자를 화면에 출력하기 위한 기능
mov     dl, bl              ; 출력할 문자를 dl에 저장
int     21h                 ; MS-DOS 호출

mov     ax, 4c00h           ; 프로그램 골격
int     21h                 ; 프로그램 골격
```

3.6 프로그램 구조

- NASM 프로그램의 컴파일, 링킹, 실행.
  - 적당한 이름으로 파일 저장(a.asm).
  - >nasm -f obj a.asm
  - a.asm을 컴파일 후 목적 파일 a.obj를 생성
  - >alink -o a a.obj
  - a.obj를 링킹 후 실행 파일 a.exe를 생성
  - >a
  - 실행 파일명만 입력하면 프로그램 실행.