# 시스템소프트웨어

## 학기 프로젝트


컴퓨터과학과

201411706 김혜지

# 1.프로젝트 계획서

The SIC or SIC/XE Assembler 설계이슈

**설계이슈 1. 어셈블러의 아키텍처를 결정한다.**

Two-pass SIC Asssembler

pass 1(SYMBOL 정의)

프로그램내의 모든 문에 주소를 배정한다.(LOCCTR)

패스 2에서 사용하기 위해 모든 레이블에 배정된 주소 값들을 저장한다.(SYMTAB)

어셈블러 지시자들에 관련된 처리를 부분적으로 행한다.(BYTE, RESW 등에 의하여 정의되는 데이타 영역의 길이 결정과 같은 주소배정에 영향을 주는 처리를 포함한다.:함수로 처리)

pass 2(명령어를 번역하고 목적 프로그램 생성)

명령어를 어셈블한다.(연산자 코드를 번역하고 주소를 조사 함.:OPTAB,SYMTAB 탐색)

BYTE, WORD 등으로 정의되는 데이타 값을 생성한다.(pass2의 지시자 처리 시 생성)

패스 1동안에 이루어지지 않는 어셈블러 지시자의 처리를 한다.(함수처리-if문 사용하여 지시자마다 각각의 처리)

목적 프로그램과 어셈블러 리스트를 출력한다.

**설계이슈 2. Micro-Instruction Set Table을 위한 데이터구조를 결정한다.**

SYMTAB: 배열 구조체

LOCCTR: 배열 구조체

OPTAB: 배열 구조체

**설계이슈 3. Micro-Instruction Set Table의 검색 알고리즘을 결정한다.**

OPTAB 탐색: 순차 탐색(for문,while 이용)

SYMTAB 탐색: 순차 탐색(단, 어드레싱 모드로 “X"가 붙었을경우 ”X"를 지운다.)

**설계이슈 4. 어셈블리언어상의 Addressing Mode 표기방법을 결정한다.**

인덱스 주소지정 방식-플래그 비트 X: 모든 주소처리를 정수로 처리하여 계산

직접주소지정 방식

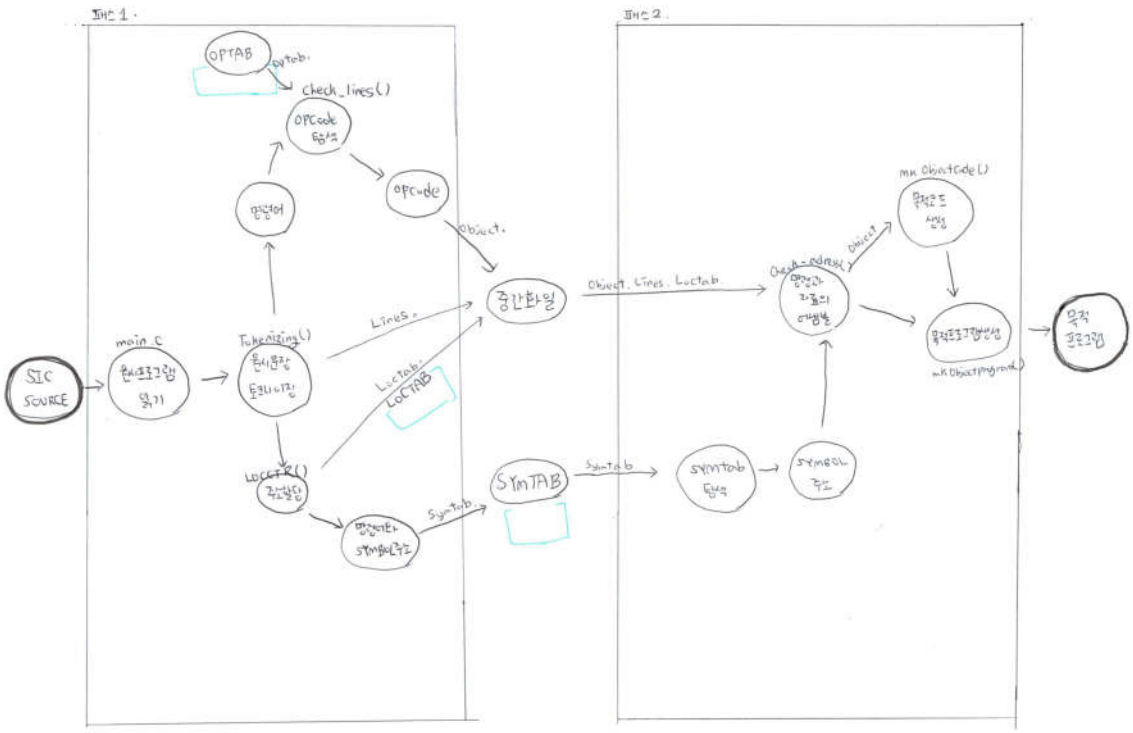
**설계이슈 5. 어셈블리언어상의 Directive 처리 방법을 결정한다.**

pass1에서 지시자를 확인하여 주소배정 후 지시자 모두 처리. BYTE는 Type이 C일 경우와 X일 경우를 나누어 목적코드 생성, WORD는 10진수를 16진수로 변환 후 목적코드 생성, RESB와 RESW는 데이터 생성 없이 메모리 예약.

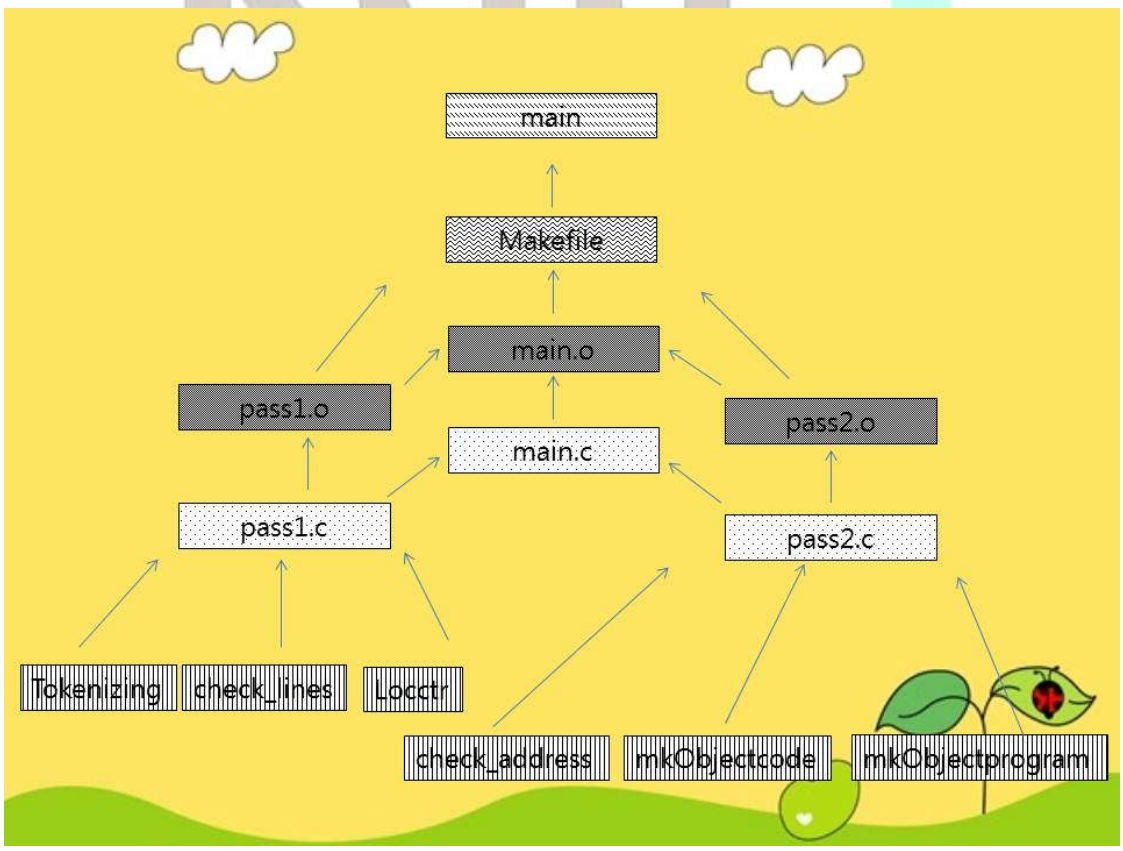
**설계이슈 6. 어셈블리 프로그램에 대한 최종 산출물인 프로그램 Object Program의 구조를 결정한다.**

목적 코드를 생성하는 함수를 만든 후, 목적프로그램 생성 함수에 H레코드, T레코트, E레코드 차례대로 생성한다.

## 2. 자료 흐름도(data flow diagram)




## 3. 프로그램 호출구조



# 4. 사용 함수 기능

- void pass1(FILE*fp,LINES*Lines,OPTAB*Optab,OBJECT*Object,SYMTAB*Symtab,LOCTAB*Loctab);
  //Tokenizing, check_lines, Locctr 함수를 호출합니다.

- void pass2(LINES*Lines,OBJECT*Object,SYMTAB*Symtab,LOCTAB*Loctab);
  //check_address,mkObjectcode,mkObjectprogram 함수를 호출합니다.

- void Tokenizing(FILE *fp,LINES *Lines);
  //어셈블러 텍스트 파일을 읽어 한 줄 당 두개 혹은 세개의 토큰을 Line[줄].token1,Line[줄].token2,Line[줄].token3에 저장합니다.

- void check_lines(LINES *Lines,OPTAB *Optab,OBJECT *Object,SYMTAB*Symtab);
  //1)Lines마다 토큰들을 체크합니다.
  //2)어떤 i번째 줄의 토큰이 Optab[j].name과 같다면 Object[i].opcode에 Optab[j].opcode를 저장하고,
  //3)같지 않다면 지시어인지 판별합니다. 만약 지시어라면 지시어대로 메모리의 크기를 Loctab.[i].Loc_countLines에 저장합니다. 이때, BYTE거나 WORD라면 OPCODE를 생성합니다.
  //4)어떤 토큰이 명령어, 지시어 둘 다 아니라면 , Symtab.Label에 저장합니다.

- void Locctr(LINES *Lines,LOCTAB *Loctab);
  //어떤 i번째 줄이라면,
  Loctab[i].Loc=Loctab[i-1]+ Loctab[i-1].Loc_count입니다.(단, 각 지시자가 가지는 메모리크기(Loctab[i].Loc_count) 처리는 check_lines함수에서 이미 하였습니다.)

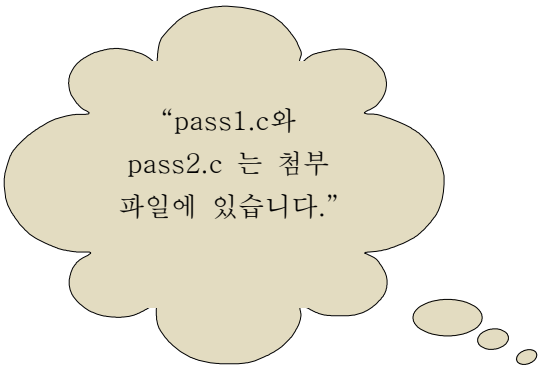
- void check_address(LINES *Lines,OBJECT *Object,SYMTAB*Symtab,LOCTAB*Loctab);
  //피연산자(token2나 token3)를 읽고,
  1)어드레싱 모드일 경우,2)어드레싱 모드가 아닐 경우 를 나눠 Symtab에서 주소를 찾아 Object.adress에 저장할 주소를 다르게 처리하였습니다.

- void mkObjectcode(LINES*Lines,OBJECT*Object);
  //Object.opcode와 Object.address를 strcat하여 목적코드를 생성하였습니다.
- void mkObjectprogram(LOCTAB*Loctab,LINES*Lines,OBJECT*Object);
  //헤더 레코드: Lines의 길이를 재서 프로그램의 길이를 구하였습니다.
  //텍스트 레코드: 목적코드 시작주소, 목적코드 길이, 목적코드 를 구하였습니다.
  //엔드 레코드

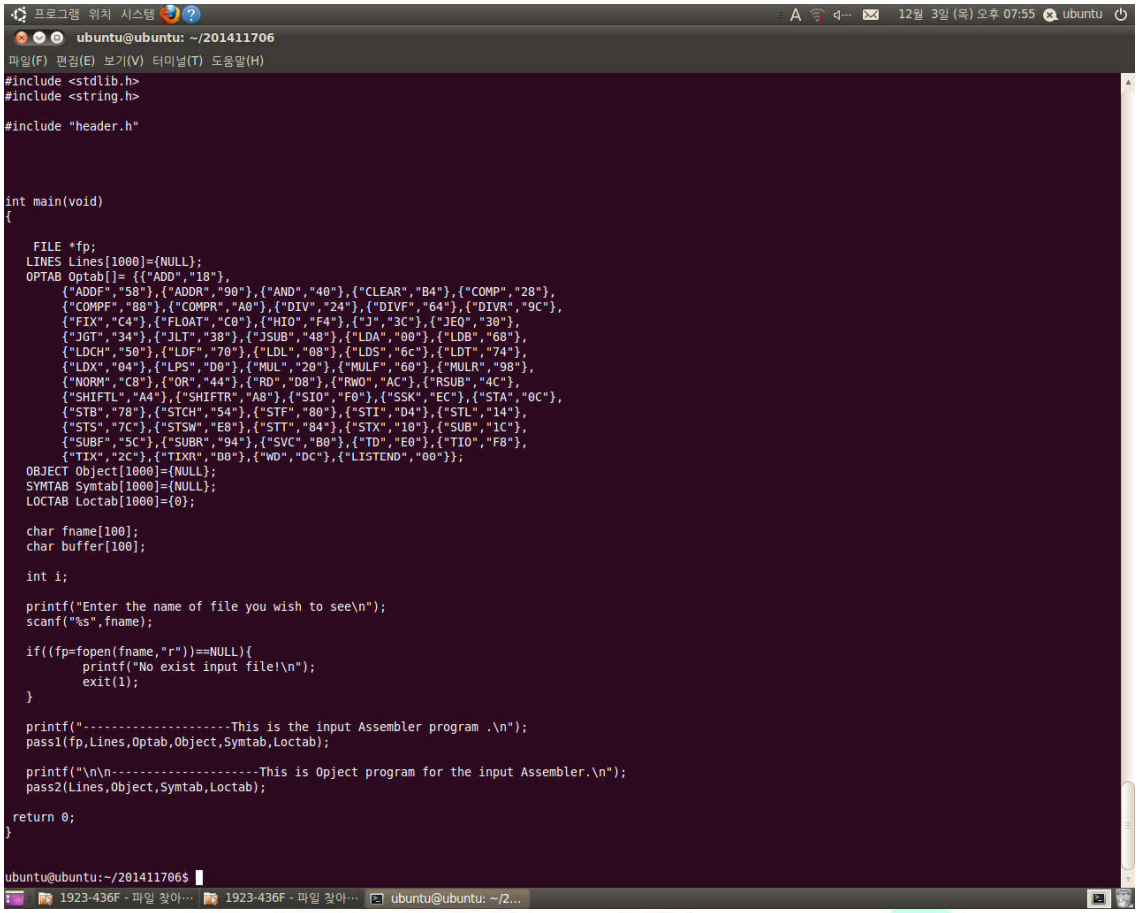
# 5. C언어로 구현한 SIC 어셈블러 소스코드

1)header.h

```
Script done, file is 201411706
ubuntu@ubuntu:~/201411706$ ./
bash: ./ : 디렉토리임
ubuntu@ubuntu:~/201411706$ ./main
Enter the name of file you wish to see
akljdsflkjasjfjl
No exist input file!
ubuntu@ubuntu:~/201411706$ cat header.h
#include <stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#define LABLE 0


typedef struct LINES
{
        char token1[10];
        char token2[10];
        char token3[10];
}LINES;

typedef struct SYMTAB
{
        char Lable[10];
        char Location[10];
}SYMTAB;
typedef struct OPTAB{
        char name[10];
        char opcode[10];
}OPTAB;
typedef struct OBJECT{
        char opcode[10];
        int address;
        char result[10];
}OBJECT;
typedef struct LOCTAB{
        int Loc;
        int Loc_count;
}LOCTAB;

void Tokenizing(FILE *fp,LINES *Lines);
void check_lines(LINES *Lines,OPTAB *Optab,OBJECT *Object,SYMTAB*Symtab,LOCTAB*Loctab);
void check_address(LINES *Lines,OBJECT *Object,SYMTAB*Symtab,LOCTAB*Loctab);

void Locctr(LINES *Lines,LOCTAB *Loctab);
void mkObjectcode(LINES*Lines,OBJECT*Object);
void mkObjectprogram(LOCTAB*Loctab,LINES*Lines,OBJECT*Object);

void pass1(FILE*fp,LINES*Lines,OPTAB*Optab,OBJECT*Object,SYMTAB*Symtab,LOCTAB*Loctab);

void pass2(LINES*Lines,OBJECT*Object,SYMTAB*Symtab,LOCTAB*Loctab);

ubuntu@ubuntu:~/201411706$
```




2)main.c


프로그램 위치 시스템
12월 3일 (목) 오후 07:55 ubuntu
ubuntu@ubuntu: ~/201411706
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
#include <stdlib.h>
#include <string.h>
#include "header.h"
int main(void)
{
FILE *fp;
LINES Lines[1000]={NULL};
OPTAB Optab[]= {{"ADD","18"},
{"ADDF","58"},{"ADDR","90"},{"AND","40"},{"CLEAR","B4"},{"COMP","28"},
{"COMPF","88"},{"COMPR","A0"},{"DIV","24"},{"DIVF","64"},{"DIVR","9C"},
{"FIX","C4"},{"FLOAT","C0"},{"HIO","F4"},{"J","3C"},{"JEQ","30"},
{"JGT","34"},{"JLT","38"},{"JSUB","48"},{"LDA","00"},{"LDB","68"},
{"LDCH","50"},{"LDF","70"},{"LDL","08"},{"LDS","6c"},{"LDT","74"},
{"LDX","04"},{"LPS","D0"},{"MUL","20"},{"MULF","60"},{"MULR","98"},
{"NORM","C8"},{"OR","44"},{"RD","D8"},{"RWO","AC"},{"RSUB","4C"},
{"SHIFTL","A4"},{"SHIFTR","A8"},{"SIO","F0"},{"SSK","EC"},{"STA","0C"},
{"STB","78"},{"STCH","54"},{"STF","80"},{"STI","D4"},{"STL","14"},
{"STS","7C"},{"STSW","E8"},{"STT","84"},{"STX","10"},{"SUB","1C"},
{"SUBF","5C"},{"SUBR","94"},{"SVC","B0"},{"TD","E0"},{"TIO","F8"},
{"TIX","2C"},{"TIXR","B8"},{"WD","DC"},{"LISTEND","00"}};
OBJECT Object[1000]={NULL};
SYMTAB Symtab[1000]={NULL};
LOCTAB Loctab[1000]={0};
char fname[100];
char buffer[100];
int i;
printf("Enter the name of file you wish to see\n");
scanf("%s",fname);
if((fp=fopen(fname,"r"))==NULL){
printf("No exist input file!\n");
exit(1);
}
printf("---------------------This is the input Assembler program .\n");
pass1(fp,Lines,Optab,Object,Symtab,Loctab);
printf("\n\n---------------------This is Opject program for the input Assembler.\n");
pass2(Lines,Object,Symtab,Loctab);
return 0;
}
ubuntu@ubuntu:~/201411706$
1923-436F - 파일 찾아…
1923-436F - 파일 찾아…
ubuntu@ubuntu: ~/2…

# 6. SIC 어셈블러 실행화면 및 결과

1)입력 파일이 없을 때.

```
ubuntu@ubuntu:~/201411706$ ./main
Enter the name of file you wish to see
akljdsflkjasjfjl
No exist input file!
ubuntu@ubuntu:~/201411706$
```

2)입력 파일(copy.txt)가 있을 때.

```
201411706  header.h      main    main.o    pass1.c   pass1.o  pass2.o
copy.txt   header.h.gch  main.c  Makefile  pass1.c~  pass2.c
hyejikim@ubuntu:~/201411706$ ./main
Enter the name of file you wish to see
copy.txt
---------------------This is the input Assembler program .
COPY    START 1000
FIRST   STL   RETADR
CLOOP   JSUB  RDREC
        LDA   LENGTH
        COMP  ZERO
        JEQ   ENDFIL
        JSUB  WRREC
        J     CLOOP
ENDFIL  LDA   EOF
        STA   BUFFER
        LDA   THREE
        STA   LENGTH
        JSUB  WRREC
        LDL   RETADR
        RSUB
EOF     BYTE  C'EOF'
THREE   WORD  3
ZERO    WORD  0
RETADR  RESW  1
LENGTH  RESW  1
BUFFER  RESB  4096
.
.       SUBROUTINE TO READ RECORD INTO BUFFER
.
        LDA   ZERO
RLOOP   TD    INPUT
        JEQ   RLOOP
        RD    INPUT
        COMP  ZERO
        JEQ   EXIT
        STCH  BUFFER,X
        TIX   MAXLEN
        JLT   RLOOP
EXIT    STX   LENGTH
        RSUB
INPUT   BYTE  X'F1'
MAXLEN  WORD  4096
.
.       SUBROUTINE TO READ RECORD INTO BUFFER
.
WRREC   LDX   ZERO
WLOOP   TD    OUTPUT
        JEQ   WLOOP
        LDCH  BUFFER,X
        WD    OUTPUT
        TIX   LENGTH
        JLT   WLOOP
        RSUB
OUTPUT  BYTE  X'05'
        END   FIRST
```

```
EXIT    STX   LENGTH
        RSUB
INPUT   BYTE  X'F1'
MAXLEN  WORD  4096
.
.       SUBROUTINE TO READ RECORD INTO BUFFER
.
WRREC   LDX   ZERO
WLOOP   TD    OUTPUT
        JEQ   WLOOP
        LDCH  BUFFER,X
        WD    OUTPUT
        TIX   LENGTH
        JLT   WLOOP
        RSUB
OUTPUT  BYTE  X'05'
        END   FIRST


---------------------This is Opject program for the input Assembler.
HCOPY  00100000107A
T0010001E141033482039001036281030301015482061 3C100300102A0C103900102D
T00101E150C10364820610810334C0000454F46000003000000
T0020391E041030001030E0205D30203FD8205D2810303020575490392C205E38203F
T0020571C1010364C0000F1001000041030E02079302064509039DC20792C1036
T002073073820644C000005
E001000
hyejikim@ubuntu:~/201411706$ exit
exit
Script done, file is 201411706
hyejikim@ubuntu:~/201411706$
```