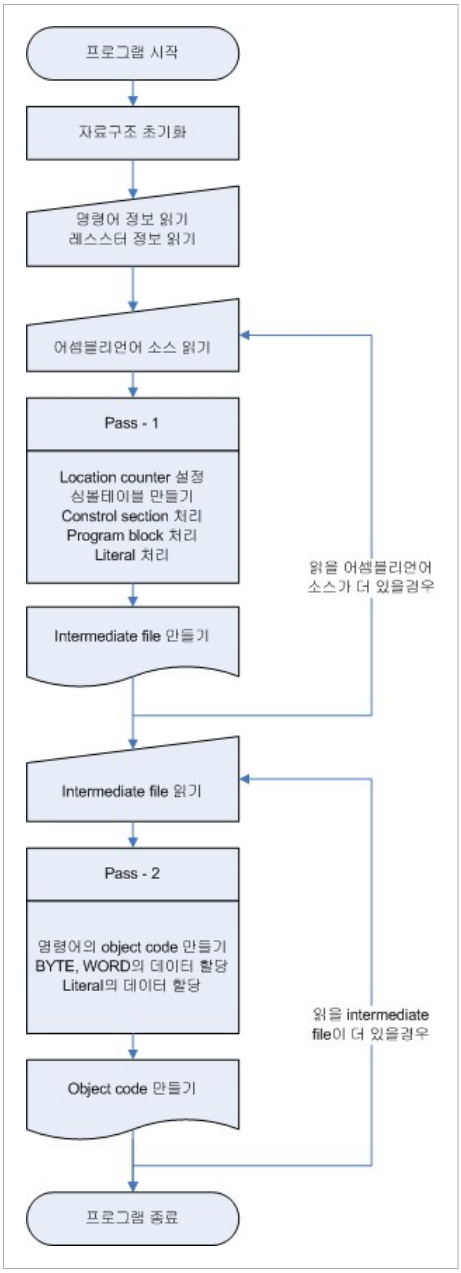
1. SIC/XE 어셈블리언어로 작성된 코드를 object 코드로 변환
  · 어셈블러를 pass 1과 pass 2 과정으로 처리
  · SIC/XE instruction set의 정보를 테이블로 구현 (OPTAB)
  · pass 1과정에서 심볼테이블 구현 (SYMTAB)
  · pass 1 과정의 출력물인 intermediate file 구현
  · 여러 가지 addressing 방법 처리 (direct, indirect, immediate)
  · 확장 format (4형식) 명령어 처리
  · literal 처리
  · program block 혹은 control section 처리
  · extern define/reference 처리
  · pass 2 과정의 출력물인 object code 만들기




```
typedef struct _inst_unit
{
    char    str[MAXLEN_INST];       // instruction name
    char    op;                     // code
    int     format;                 // format
    int     ops;                    // count of operand
} inst_unit;


typedef struct _sym_unit
{
    char    symbol[MAXLEN_LABEL];   // label
    int     value;                  // address
    int     blk_num;                // 해당 block number
} sym_unit;

typedef struct _reg_unit
{
    char    name[MAXLEN_REGNAME];   // regsiter name
```

```
    int     num;                    // register number
} reg_unit;


OPTAB   optab;          // operation code table
SYMTB   symtab;         // symbol table
INTER   inter;          // intermediate file

main()
{
    int locctr;         // location counter

    init_reg(...);      // initialize information of register
    init_inst(...);     // initialize information of instruction

    pass1();
    pass2();
}

pass1()
{
    insert_csect(...);      // insert a new control section
    insert_block(...);      // insert a new program block

    search_opcode(...);     // search opcode from table

    insert_symbol(...);     // insert symbol into symbol table

    insert_inter(...);      // insert a new intermediate source

    insert_literal(...);    // insert a literal to literal pool
}

pass2()
{
    .
    .
    .
    gen_objcode(op, n, i, x, b, p, e, disp, format, code);  // generate object code
    .
    .
    .
}

// generate object code
gen_objcode(op, n, i, x, b, p, e, disp, format, code)
{
    switch(format)
    {
        case 1:     // format 1
            sprintf(code, "%02X", op);
            break;

        case 2:     // format 2
            sprintf(code, "%02X%1X%1X", op, n, l);
            break;

        case 3:     // format 3
        case 4:     // format 4
            .
            .
            .
    }
}

// 10진수 문자를 숫자 값으로
int char_to_decnum(const char *str)
{
    int sum = 0;

    while(*str != '₩0')
```

```
    {
        if(isdigit(*str) == 0)
            break;
        sum *= 10;
        sum += *str - '0';
        str++;
    }

    return sum;
}

// 16진수 문자를 숫자 값으로
int char_to_hexnum(const char *str)
{
}

// 입력 문자열로 명령어 정보 얻기
inst_unit *search_opcode(const char *str)
{
}

// header record 설정
set_header_record(...)
{
}

// extern define record 설정
set_extdef_record(...)
{
}

// extern reference record 설정
set_extref_record(...)
{
}

// text record 설정
set_text_record(...)
{
}

// modify record 설정
set_modify_record(...)
{
}

// end record 설정
set_end_record(...)
{
}

// expression 계산
int calc_expr(...)
{
}
```