시스템 프로그래밍 Project #1

* **과제내용**
  - ControlSection 방식의 SIC/XE 소스를 Object Program Code로 바꾸는 어셈블러 만들기
  - SIC/XE 소스를 라인별로 처리해서 Object Code로 바꾼 후, Object Program Code로 변환하는 프로그램
  - 과제에 주어진 C 소스코드와 헤더 파일 사용하기

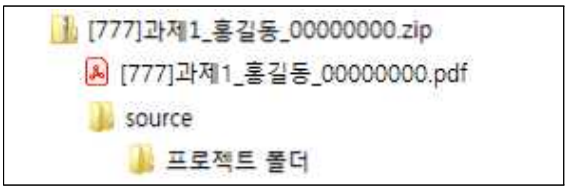
* **과제 목적**
  - SIC/XE 소스를 Object Program Code로 변환해봄으로써 SIC/XE 어셈블러의 동작을 이해한다.
  - 주어진 C 소스코드 외 헤더파일을 이용하여 SIC/XE 소스를 Object Program Code로 변환하는 과정을 이해하고 이후 확장되는 과제 내용에 맞추어 프로그램의 확장성을 효과적으로 증진시키기 위한 기본 지식을 학습한다.

* **과제 제출 마감 -** 4월 29일(월) 오전 11:59까지 스마트 캠퍼스 과제란에 제출
  (제출시간 이후 제출 시 10 point 패널티 부과, 이후부터는 매일 10 point씩 추가 패널티를 부과함)

* **제출물 - 레포트 파일(PDF)** + 프로그램 소스코드(.c 및 .h 파일) + 입력 파일(input.txt 및 inst_table.txt 파일) + 출력 결과 파일(output_symtab.txt, output_littab.txt, output_objectcode.txt 파일)

* **제출 레포트 (50 point)**
  - 요구사항 : 미니 커뮤니티 혹은 스마트 캠퍼스에 올라간 보고서 양식을 사용할 것 (5p) (학번, 이름, 출석번호, 과제명, 수업 구분<가,나>)
    1. 동기/목적(5p) 2. 설계/구현 아이디어(10p) 3. 수행결과(10p)
    4. 결론 및 보충할 점(10p) 5. 소스코드(+주석)(10p)
  - 소스코드는 2단으로 출력할 것 (별도로 아래와 같이 파일로도 제출)
  - 점수 평가에서 레포트의 비중이 높으므로 제출 마감 전까지 성심껏 작성하기 바랍니다.

* **제출 파일양식 (50 point) -** [출석번호]프로젝트1_이름_00000000.zip
  - 레포트 파일은 PDF로 한정
  - 소스코드는 프로젝트 폴더를 그대로 첨부
    - 프로젝트 폴더 내에 소스코드, 입력 파일, 출력 결과 파일이 존재해야 함
  - 소스코드의 “00000000” 은 자신의 학번으로 교체



(제출 파일 구성 예시)

* 제출 파일양식을 지키지 않을 시 미제출로 간주
* 제출 파일은 smart-campus 과제게시판에 올릴 것
* 기간 내 레포트 및 파일 미제출 시 Late Penalty 부여
* 프로그램 Input/Output은 표시된 Input/Output 문서를 기준으로 함

*** 프로그램 구현에 사용해야 할 인터페이스 내용**

- 매핑을 위한 OPCODE 테이블은 Appendix 참고하여 직접 작성한다.
- 이전 과제에서 주석처리 되어있던 함수들을 사용한다.
- 아래에 주어진 명세를 참고하여 과제 코드를 구현할 것

1. 기계어 목록 관리를 위한 구조체 정보

```
/** SIC/XE 머신의 instruction 정보를 저장하는 테이블 */
inst *inst_table[MAX_INST_TABLE_LENGTH];
int inst_table_length;

/**
 * @brief 한 개의 SIC/XE instruction을 저장하는 구조체
 *
 * @details
 * 기계어 목록 파일(inst_table.txt)에 명시된 SIC/XE instruction 하나를
 * 저장하는 구조체. 라인별로 하나의 instruction을 저장하고 있는 instruction 목록
 * 파일로부터 정보를 받아와서 생성한다.
 */
typedef struct _inst {
    char str[10];     /** instruction 이름 */
    unsigned char op; /** instruction의 opcode */
    int format;       /** instruction의 format */
    int ops;          /** instruction이 가지는 operator 개수 */
} inst;
```

2. input 파일의 관리를 위한 구조체 정보

```
/** SIC/XE 소스코드를 저장하는 테이블 */
char *input[MAX_INPUT_LINES];
int input_length;

/**
 * @brief 소스코드 한 줄을 분해하여 저장하는 구조체
 *
 * @details
 * 원할한 assem을 위해 소스코드 한 줄을 label, operator, operand, comment로
 * 파싱한 후 이를 저장하는 구조체. 필드의 `operator`는 renaming을 허용한다.
 */
typedef struct _token {
    char *label;   /** label을 가리키는 포인터 */
    char *operatr; /** operator를 가리키는 포인터 */
    char *operand[MAX_OPERAND_PER_INST]; /** operand들을
                                          가리키는 포인터 배열 */
    char *comment; /** comment를 가리키는 포인터 */
    char nixbpe;   /** 특수 bit 정보 */
} token;

/** 소스코드의 각 라인을 토큰 전환하여 저장하는 테이블 */
token *tokens[MAX_INPUT_LINES];
int tokens_length;
```

3. 소스코드 내의 symbol 및 literal 관리를 위한 구조체 정보

```
/** 소스코드 내의 심볼을 저장하는 테이블 */
symbol *symbol_table[MAX_TABLE_LENGTH];
int symbol_table_length;

/**
 * @brief 하나의 심볼에 대한 정보를 저장하는 구조체
 *
 * @details
 * SIC/XE 소스코드에서 얻은 심볼을 저장하는 구조체이다. 기존에 정의된 `name` 및
 * `addr`는 필수로 사용해야 한다. 필드가 더 필요한 경우 구조체 내에 필드를
 * 추가하는 것을 허용한다.
 */
typedef struct _symbol {
    char name[10]; /** 심볼의 이름 */
    int addr;      /** 심볼의 주소 */
    /* add fields if needed */
} symbol;

/** 소스코드 내의 리터럴을 저장하는 테이블 */
literal *literal_table[MAX_TABLE_LENGTH];
int literal_table_length;

/**
 * @brief 하나의 리터럴에 대한 정보를 저장하는 구조체
 *
 * @details
 * SIC/XE 소스코드에서 얻은 리터럴을 저장하는 구조체이다. 기존에 정의된 literal
 * 및 addr는 필수로 사용하고, field가 더 필요한 경우 구조체 내에 field를
 * 추가하는 것을 허용한다. addr 필드는 리터럴의 값을 저장하는 것이 아닌 리터럴의
 * 주소를 저장하는 필드임을 유의하라.
 */
typedef struct _literal {
    char literal[20]; /** 리터럴의 표현식 */
    int addr;         /** 리터럴의 주소 */
    /* add fields if needed */
} literal;
```

4. 오브젝트 코드를 관리하기 위한 구조체 정보

```
/** 오브젝트 코드를 저장하는 변수 */
object_code *obj_code = NULL;

/**
 * @brief 오브젝트 코드 전체에 대한 정보를 담는 구조체
 *
 * @details
 * 오브젝트 코드 전체에 대한 정보를 담는 구조체이다. Header Record, Define
 * Record, Modification Record 등에 대한 정보를 모두 포함하고 있어야 한다. 이
 * 구조체 하나만으로 object code를 충분히 작성할 수 있도록 구조체를 직접
 * 정의해야 한다.
 */
typedef struct _object_code {
    /* add fields */
} object_code;
```

- my_assembler_00000000.c 파일에 명시된 함수들을 구현하고, 추가적으로 필요한 함수 구현과 변수 생성은 자유.
  (단, 기본 함수는 모두 사용해야 함, 추가한 함수는 레포트에 사용 목적을 명시할 것)

*** 프로그램 수행에 따른 입력과 출력은 다음과 같아야 한다.**

input.txt (공백 문자를 변경하여 포맷을 바꾸는 행위는 허용함)

```
COPY  START   0        COPY FILE FROM IN TO OUTPUT
      EXTDEF  BUFFER,BUFEND,LENGTH
      EXTREF  RDREC,WRREC
FIRST STL     RETADR   SAVE RETURN ADDRESS
CLOOP +JSUB   RDREC    READ INPUT RECORD
      LDA     LENGTH   TEST FOR EOF (LENGTH = 0)
      COMP    #0
      JEQ     ENDFIL   EXIT IF EOF FOUND
      +JSUB   WRREC    WRITE OUTPUT RECORD
      J       CLOOP    LOOP
ENDFIL        LDA     =C'EOF'  INSERT END OF FILE MARKER
      STA     BUFFER
      LDA     #3       SET LENGTH = 3
      STA     LENGTH
      +JSUB   WRREC    WRITE EOF
      J       @RETADR  RETURN TO CALLER
RETADR        RESW    1
LENGTH        RESW    1        LENGTH OF RECORD
      LTORG
BUFFER        RESB    4096     4096-BYTE BUFFER AREA
BUFEND        EQU     *
MAXLEN        EQU     BUFEND-BUFFER     MAXIMUM RECORD LENGTH
RDREC CSECT
.
.     SUBROUTINE TO READ RECORD INTO BUFFER
.
      EXTREF  BUFFER,LENGTH,BUFEND
      CLEAR   X        CLEAR LOOP COUNTER
      CLEAR   A        CLEAR A TO ZERO
      CLEAR   S        CLEAR S TO ZERO
      LDT     MAXLEN
RLOOP TD      INPUT    TEST INPUT DEVICE
      JEQ     RLOOP    LOOP UNTIL READY
      RD      INPUT    READ CHARACTER INTO REGISTER A
      COMPR   A,S      TEST FOR END OF RECORD (X'00')
      JEQ     EXIT     EXIT LOOP IF EOR
      +STCH   BUFFER,X STORE CHARACTER IN BUFFER
      TIXR    T        LOOP UNLESS MAX LENGTH
      JLT     RLOOP    HAS BEEN REACHED
EXIT  +STX    LENGTH   SAVE RECORD LENGTH
      RSUB             RETURN TO CALLER
INPUT BYTE    X'F1'    CODE FOR INPUT DEVICE
MAXLEN        WORD     BUFEND-BUFFER
WRREC CSECT
.
.     SUBROUTINE TO WRITE RECORD FROM BUFFER
.
      EXTREF  LENGTH,BUFFER
      CLEAR   X        CLEAR LOOP COUNTER
      +LDT    LENGTH
WLOOP TD      =X'05'   TEST OUTPUT DEVICE
      JEQ     WLOOP    LOOP UNTIL READY
      +LDCH   BUFFER,X GET CHARACTER FROM BUFFER
      WD      =X'05'   WRITE CHARACTER
      TIXR    T        LOOP UNTIL ALL CHARACTERS
      JLT     WLOOP    HAVE BEEN WRITTEN
      RSUB             RETURN TO CALLER
      END     FIRST
```

output_symtab.txt (패스 1 종료 후 출력. 포맷은 중요하지 않음)

```
COPY    0
FIRST   0           +1 COPY
CLOOP   3           +1 COPY
ENDFIL  17          +1 COPY
RETADR  2A          +1 COPY
LENGTH  2D          +1 COPY
BUFFER  33          +1 COPY
BUFEND  1033        +1 COPY
MAXLEN  1000
RDREC   0
RLOOP   9           +1 RDREC
EXIT    20          +1 RDREC
INPUT   27          +1 RDREC
MAXLEN  28          +1 RDREC
WRREC   0
WLOOP   6           +1 WRREC
```

output_littab.txt (패스 1 종료 후 출력. 포맷은 중요하지 않음)

```
=C'EOF'      30
=X'05'       1B
```

output_objectcode.txt (패스 2 종료 후 출력. 출력 결과는 아래 예시와 반드시 *동일해야 함. 각 라인 뒤쪽의 공백 문자 혹은 개행 문자의 차이는 허용함)

```
HCOPY  000000001033
DBUFFER000033BUFEND001033LENGTH00002D
RRDREC WRREC
T0000001D1720274B10000003202329000033200074B1000003F2FEC0320160F2016
T00001D0D0100030F200A4B1000003E2000
T00003003454F46
M00000405+RDREC
M00001105+WRREC
M00002405+WRREC
E000000
HRDREC 00000000002B
RBUFFERLENGTHBUFEND
T0000001DB410B400B44077201FE3201B332FFADB2015A00433200957900000B850
T00001D0E3B2FE9131000004F0000F1000000
M00001805+BUFFER
M00002105+LENGTH
M00002806+BUFEND
M00002806-BUFFER
E
HWRREC 00000000001C
RLENGTHBUFFER
T0000001CB41077100000E32012332FFA53900000DF2008B8503B2FEE4F000005
M00000305+LENGTH
M00000D05+BUFFER
E
```