2025 시스템 프로그래밍 Project #1

* **과제내용**
  - ControlSection 방식의 SIC/XE 소스(Fig 2.15)를 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

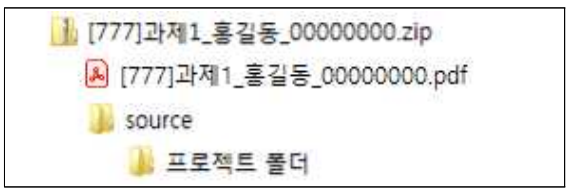
* **과제 목적**
  - SIC/XE 소스를 Object Program Code 로 변환하여 봄으로써 SIC/XE 어셈블러의 동작을 이해한다.
  - 주어진 C 소스코드 외 헤더파일을 이용하여 SIC/XE 소스를 Object Program Code로 변환하는 과정을 이해하고 이 후 확장되는 과제 내용에 맞추어 프로그램의 확장성을 효과적으로 증진시키기 위한 기본 지식을 학습한다.

* **과제 제출 마감 - 4/25 (금) 오후 11:59까지 스마트캠퍼스에 제출**
  (제출시간 이후에는 매일 10 point씩 추가 패널티를 부과함)

* **제출물 – 레포트 파일(PDF, hwp, doc) + 프로그램 소스코드**

* **제출 레포트 (50 point)**
  - 요구사항 : 미니 커뮤니티 혹은 스마트 캠퍼스에 올라간 보고서 양식을 사용할 것 (5p)
    1. 동기/목적(5p) 2. 설계/구현 아이디어(10p) 3. 수행결과(10p)
    4. 결론 및 보충할 점(10p) 5. 소스코드(+주석)(10p)

* **제출 파일양식 (50 point) – [출석번호]프로젝트1_이름_(학번).zip**
  - 레포트 파일은 PDF로 한정
  - 소스코드는 프로젝트 폴더를 그대로 첨부
    - 비주얼 스튜디오의 경우 .sln 파일 반드시 첨부
  - 소스코드의 “00000000”위치는 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 사용해야할 인터페이스 내용**

- 매핑을 위한 OPCODE 테이블은 Appendix 참고하여 직접 작성한다.
- 아래에 주어진 명세를 참고하여 과제 코드를 구현할 것

파일명 : my_assembler_00000000.h
내용 : my_assembler_00000000.c를 위한 변수선언과 테이블 관리를 위한 구조체 생성

1. input 프로그램 관리를 위해 기술되는 구조체 정보

```
// 어셈블리 할 소스코드를 파일로부터 불러와 라인별로 관리하는 테이블 생성
char *input_data[MAX_LINES];
static int line_num;

// 어셈블리 할 라인 별 소스코드를 토큰 단위로 관리하기 위한 구조체 변수
struct token_unit {
       char *label;                    // 명령어 라인 중 label
       char *operator;                 // 명령어 라인 중 operator
       char *operand[MAX_OPERAND];     // 명령어 라인 중 operand
       char comment[100];              // 명령어 라인 중 comment
      char nixbpe;                     // 하위 6bit 사용 : _ _ n i x b p e
};

//  어셈블리 할 소스코드를 5000라인까지 관리하는 테이블 생성
typedef struct token_unit token;
token *token_table[MAX_LINES];
```

2. Input 파일의 소스코드 중 symbol 영역에 해당하는 token을 관리하기 위한 구조체 생성 및 구조체 변수의 배열 선언

```
// symbol에 해당하는 token의 정보를 기술하는 구조체 변수
// symbol의 이름과 주소 정보를 가지고 있다.
struct symbol_unit {
            char symbol[10];
            int addr;
};

// symbol token을 관리하는 테이블 생성
typedef struct symbol_unit symbol;
symbol sym_table[MAX_LINES];
```

3. Input 파일의 소스코드 중 literal 영역에 해당하는 token을 관리하기 위한 구조체 생성 및 구조체 변수의 배열 선언

```
// literal에 해당하는 token의 정보를 기술하는 구조체 변수
// literal의 이름과 주소 정보를 가지고 있다.
struct literal_unit{
            char *literal;
            int addr;
}

// literal token을 관리하는 테이블 생성
typedef struct literal_unit literal;
literal literal_table[MAX_LINES];
```

4. 파서 과제 (과제 6번)에서 변경 사항

```
int main(int args, char *arg[]) {
	if(init_my_assembler()< 0) {
		printf("init_my_assembler: 프로그램 초기화에 실패 했습니다.\n");
		return -1 ;
	}

	if(assem_pass1() < 0 ){		// pass1 나머지 과정 추가 구현
		printf("assem_pass1: 패스1 과정에서 실패하였습니다.  \n") ;
		return -1 ;
	}

	/* opcode 출력은 삭제*/
	// make_opcode_output("output_00000000");

	/* 새로이 구현되는 내용 */
	make_symtab_output("symtab_00000000");		// symbol table 출력
	make_literaltab_output("literaltab_00000000");	// literal table
	if(assem_pass2() < 0 ){				// object code 생성
		printf(" assem_pass2: 패스2 과정에서 실패하였습니다.  \n") ;
		return -1 ;
	}

	make_objectcode_output("output_00000000");	// object code 출력

	return 0;
}

void make_symtab_output(char *file_name){
	/* Symbol Table 값 16진수로 출력 */
}

void make_literaltab_ouput(char *file_name){
	/* Literal Table 값 16진수로 출력 */
}
```

- my_assembler.c 파일에 이미 명시되어있는 함수들을 구현하고,
  추가적으로 필요한 함수 구현과 변수 생성은 자유
  (단, 기본 함수는 모두 사용해야 함, 리포트에 사용 목적을 명시할 것)

* 프로그램 수행에 따른 입력과 출력은 다음과 같아야 한다.

Input

```
        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'  I               NSERT 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
```

pass1 종료 후 Output(화면 출력) *(정렬은 중요하지 않음)*

- symtab

```
COPY                    0
FIRST                   0
CLOOP                   3
ENDFIL                  17
RETADR                  2A
LENGTH                  2D
BUFFER                  33
BUFEND                  1033
MAXLEN                  1000

RDREC                   0
RLOOP                   9
EXIT                    20
INPUT                   27
MAXLEN                  28

WRREC                   0
WLOOP                   6
```

- literaltab

```
EF                      30
05                      1B
```

Output(파일 출력)

```
HCOPY  000000001033
DBUFFER000033BUFEND001033LENGTH00002D
RRDREC WRREC
T0000001D1720274B10000003202329000003320074B1000003F2FEC0320160F2016
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
```