



# 시스템소프트웨어와 실습

## Project1.

제출일	2023년 11월 14일	학과	컴퓨터공학과
과목	시소프 1분반	학번	2022111967
담당교수	정준호 교수님	이름	조현우

## 소스 코드 설명

- **Assembler.h** (어셈블러에서 사용할 함수를 정의)

```
#pragma once
#include<iostream>
#include<fstream>
#include<string>
#include<sstream>
#include<unordered_map>
#include<map>
using namespace std;
#define endl "\n"

// Functions.cpp
void Make_Optable(unordered_map<string, string>& optable);
void Update_Locctr(string opcode, string operand, int& LOCCTR);
void classify_command(string line, string& label, string& opcode, string& operand);
void classify_intfile_command(string line, string& label, string& opcode, string& operand);
string intToHexString(int value, int length);
void change_upper(string& line);

// 2Pass.cpp
void Pass1();
void Pass2();

//Additional_Function.cpp
bool check_litteral(string operand);
void addLitteral(string& operand, int LOCCTR, ofstream& intfile);
void InputLitteralToIntfile(ofstream& intfile);
string FindLitteralFromLittab(string operand);
```

(함수에 대한 설명은 아래 cpp파일 설명에서 설명.)

- **Functions.cpp**

### (Make\_optable)

파일로 저장된 미모닉 코드와, 머신 코드를 읽어 OPTAB을 만드는 함수이다. 해쉬 테이블을 이용하여, 미모닉코드를 넣으면 머신코드가 출력될 수 있게 하였다.

### (Update\_Locctr)

Opcode에 알맞게 Locctr를 업데이트 해주는 함수로, word, byte, resw, resb, 나머지로 나누어 Locctr를 업데이트한다.

### (classify\_command)

srcfile에서 한 줄 씩 읽었을 때, label, opcode, operand를 구분하고 comment는 버리게 해주는 함수이다.

### (classify\_intfile\_command)

위 함수와 같은 작업을 intfile에서 실행한다, intfile은 \t으로 구분할 것이기 때문에, 따로 함수를 만들었다.

### **(intToHexString)**

LOCCTR를 start가 나왔을 때 16진수로 읽어 10진수로 저장한 후, 업데이트하며 저장해 나갈 것이다. 그렇기 때문에 10진수인 int를 16진수로 바꿔주는 함수를 구현하였다.

### **(change\_upper)**

소문자를 대문자로 변환해주는 함수를 구현하였다.

## ● **2PassAlgorithm.cpp**

2pass-algorithm에서 사용할 optab, symtab, locctr, startingaddress를 전역변수로 선언하였다.

### **(Pass1)**

input으로 srcfile를, output으로 intfile을 설정하였다.

1. Make\_optable 함수로 optab를 만든다.
2. srcfile에서 가져올 문장이 없을 때까지 한 줄씩 가져온다.
3. 해당 문장을 label, opcode, operand로 구별하며, start일 경우 locctr, startingaddress를 초기화하며, end가 나오면 반복문을 종료한다.
4. Intfile를 작성하며, label이 있을 경우엔 symtab도 작성한다.
5. Optab에 저장되지 않았을 경우 매크로인지 확인하며, 있을 경우 해당 매크로 내용을 실행함
6. 모두 아니면 오류 일으킴.

### **(Pass2)**

input으로 pass1에서 만들어진 intfile을 사용하며, output으로 objfile을 만듦

1. Intfile을 한 줄씩 읽어가며 label, opcode, operand를 구분함.
2. Opcode와, record의 길이에 따라서 record를 objfile에 입력 후 record를 초기화함. (태그도 opcode에 따라 수정)
3. Optab와 symtab에서 코드들을 가져와 6자리 명령어를 만들어 record에 저장.
4. 6자리가 안된다면 0을 채움.

- 추가구현 (Additional\_Function.cpp)

**(check\_literal)**

Operand가 저장되어 있는 리터럴인지 체크해주는 함수로, LITTAB의 name(label)을 검사한다.

**(addLiteral)**

Pass1 과정에서 =value 이런 식으로 있는 operand를 리터럴의 name으로 바꿔주며, LITTAB에 리터럴을 저장하는 함수이다.

**(InputLiteralToIntfile)**

Intfile을 만들 때 사용하는 함수로, LITTAB에 저장되어 있는 리터럴들을 intfile에 저장하고, LITTAB를 초기화 한다.

**(FindLiteralFromLittab)**

Pass2 과정에서 리터럴의 name에 따라 value를 리턴해주는 함수로, operand와 name이 같은 리터럴을 LITTAB에서 찾아 value를 리턴한다. (word일 경우 4자리, byte일 경우 한 글자 당 2 자리씩 추가하여 리턴한다)

## 결과 분석

### 1. 어셈블러 기능 구현

#### ● Input

test5	start	1000	
first	j	begin	
five	word	5	
xxx	resw	1	
begin	lda	zero	
	ldx	zero	
loop	jsub	getc	
	sta	copys	rmo a,s
	mul	sixteen	shifl s,4
	sta	copys	shifl s,4
	jsub	getc	
	add	copys	addr s,a
	stch	exaddr,x	tixr t
	tix	twelve	
	j	loop	
	stx	temp	
getc	ldx	incnt	
	lda	incnt	
	add	one	
	sta	incnt	
	ldch	inde v,x	
	comp	eof	
	jeq	exaddr	
	comp	fourtye	
	jlt	getc	
	sub	fourtye	
	comp	ten	
	jlt	return	
	sub	seven	
	comp	sixteen	
	jgt	getc	
return	ldx	temp	
	rsub		
zero	word	0	
one	word	1	
seven	word	7	
ten	word	10	
twelve	word	12	
sixteen	word	16	
fourtye	word	48	
eof	word	63	
copys	resw	1	
	resw	1	
incnt	word	0	
inde v	byte	c'0010030C1006'	
	byte	c'000005FFFFFFF'	
exaddr	resb	12	
	end	first	

```
add 18
and 40
clear b4
comp 28
div 24
fix c4
float c0
hio f4
j 3c
jeq 30
jgt 34
jlt 38
jsub 48
lda 00
ldch 50
ldx 04
mul 20
or 44
rsub 4c
sta 0c
stch 54
stl 14
stx 10
sub 1c
td e0
tio f8
tix 2c
wd dc
```

(srcfile.txt

optab.txt)

#### ● Output

LABEL	ADDRESS	LABEL	OPCODE	OPERAND
test5	1000	test5	start	1000
first	1003	first	j	begin
five	1006	five	word	5
xxx	1009	xxx	resw	1
begin	100f	begin	lda	zero
getc	102a	loop	ldx	zero
return	105a		jsub	getc
zero	1060		sta	copys
one	1063		mul	sixteen
seven	1066		sta	copys
ten	1069		jsub	getc
twelve	106c		add	copys
sixteen	106f		stch	exaddr,x
fourtye	1072		tix	twelve
eof	1075		j	loop
copys	1078	getc	stx	temp
temp	107b		ldx	incnt
incnt	107e		lda	incnt
inde v	1081		add	one
exaddr	109a		sta	incnt
			ldch	inde v,x
			comp	eof
			jeq	exaddr
			comp	fourtye
			jlt	getc
			sub	fourtye
			comp	ten
			jlt	return
			sub	seven
			comp	sixteen
			jgt	getc
			ldx	temp
		return	rsub	
			ldx	
		zero	word	0
		one	word	1
		seven	word	7
		ten	word	10
		twelve	word	12
		sixteen	word	16
		fourtye	word	48
		eof	word	63
		copys	resw	1
		temp	resw	1
		incnt	word	0
		inde v	byte	c'0010030C1006'
			byte	c'000005FFFFFFF'
		exaddr	resb	12
			end	first

(SYMTAB.txt

INTFILE.txt )

```
HTEST5 0010000000A6
T001000063C1009000005
T0010091E00106004106048102A0C107820106F0C107848102A18107854909A2C106C
T0010271E3C100F10107B04107E00107E1810630C107E50908128107530109A281072
T0010451E38102A1C107228106938105A1C106628106F34102A04107B4C0000000000
T0010631500000100000700000A00000C00001000003000003F
T00107E1C00000030303130303330433130303630303030303546464646463F
E001000
```

(OBJFILE.txt)

```
HTEST5 0010000000A6
T001000063C1009000005
T0010091E00106004106048102A0C107820106F0C107848102A18107854909A2C106C
T0010271E3C100F10107B04107E00107E1810630C107E50908128107530109A281072
T0010451E38102A1C107228106938105A1C106628106F34102A04107B4C0000000000
T0010631500000100000700000A00000C00001000003000003F
T00107E1C00000030303130303330433130303630303030303546464646463F
E001000
```

(위는 Dosbox에서 생성된 OBJFILE 내용)

SYMTAB, INTFILE, OBJFILE를 출력하였으며, dosbox에서 만들어진 objfile의 내용과 같다는 것을 확인할 수 있기 때문에 정상적으로 2PassAlgorithm이 작동한다.

## 2. 추가구현 (리터럴)

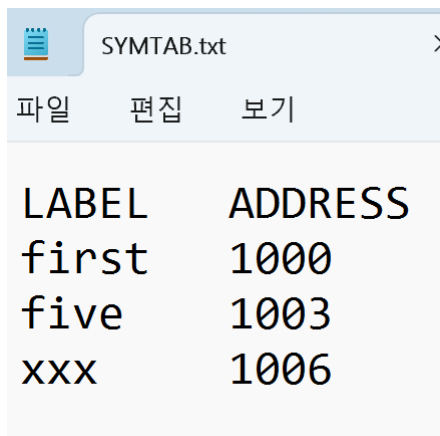
- Input

```
test5    start    1000
first    j        begin
five     word     5
xxx      resw     1
begin    lda      =0
          ldx      =c'zero'
          end      first
```

(srcfile.txt),

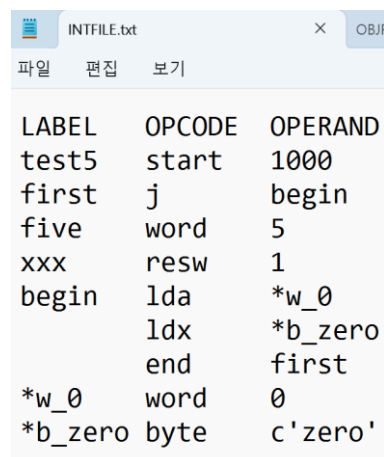
(optab.txt는 1번 예시와 동일)

- Output



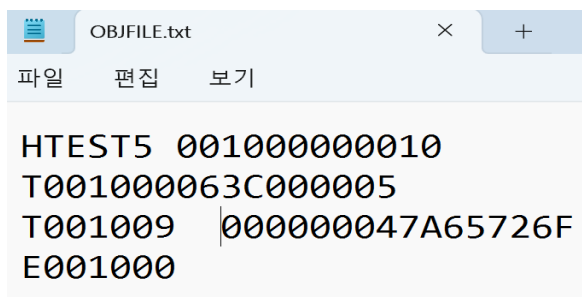
LABEL	ADDRESS
first	1000
five	1003
xxx	1006

(SYMTAB.txt



LABEL	OPCODE	OPERAND
test5	start	1000
first	j	begin
five	word	5
xxx	resw	1
begin	lda	*w_0
	ldx	*b_zero
	end	first
*w_0	word	0
*b_zero	byte	c'zero'

INTFILE.txt)



```
HTEST5 001000000010
T001000063C000005
T001009 |000000047A65726F
E001000
```

(OBJFILE.txt)

objfile에서 리터럴의 길이는 표현하지 못했지만, =0 에서 000000 이 들어갔으며, =c'zero'에선 정  
상적으로 7A65726F가 들어갔다, 또한 intfile에서는 end와 만나 그 아래 리터럴들이 생성되었다.