Computer Science & Engineering Department

I. I. T. Kharagpur

Compiler Lab (CS39003)

3nd Year CSE 5th Semester

Assignment: Implement an assembler in C

Submission Deadline: 8th August 2013, Midnight. **Marks: 15**

Important Instructions:

In this assignment, you are asked to write a two pass assembler in C. Your assembler module should take the SIC assembly language program as input and generate the object code in the specified format (available in the study material). You may assume that one Text record in the object code is sufficient to contain the entire translated instructions and data. So there will be only one Text record in the translated object code.

Things you need to do in precise:

i. Understand the pesudocode for PASS I and PASS II of the Assembler that was discussed in class.

ii. Write a **C program (mandate)** to have your own PASS I and PASS II codes. Thus you have an assembler code composed of maybe passI.c and passII.c

iii. The input to passI.c should be one of the assignment SIC program files (given below) [e.g., Celcius-Farenheit.txt].

iv. The output of passI.c should produce the intermediate code in the format described to you in class [e.g., CF-intermediate.txt].

v. Please note that each entry the intermediate code should contain the source assembly language instruction along with the location counter value and error messages (if applicable).

Format of the intermediate representation

Location counter **Source instruction** **Error message (if exist)**

vi. The input to the passII.c should be the intermediate codes: [e.g., CF-intermediate.txt].

vii. The output of passII.c should produce the corresponding object code [e.g., CF-object.txt].

What to submit for evaluation:

i. Assembler codes [e.g., passI.c and passII.c]

ii. Intermediate codes for each source program [e.g., CF-intermediate.txt].

iii. Object codes for each source program [e.g., CF-object.txt].

iv. A read me file

How we will evaluate:

i. We shall run passI.c with all assembly language programs.

ii. We shall run passII.c with all intermediate codes as inputs.

iii. We shall check for correctness of all object codes.

iv. We shall check for valid error messages in case of error contain in the assembly language programs.

Note that we are not going to dump your object code on any processor because we simply do not have an SIC compliant processor. So you really do not worry which memory location your program is going to start from as you will never attempt an actual access to that memory.

Problems:

Generate object codes for SIC machine using Two-pass Assembler for the following assembly language programs. Note that these are only sample source programs. Your source should be able to translate any assembly language program written in SIC language (and report proper error messages, where applicable).

The syntax information is available in the study material uploaded on Intinno or on http://cse.iitkgp.ac.in/~bivasm/teaching.html. The machine opcode table is provided. Assume each memory address is of 2 bytes long. Memory addresses are expressed in Hexadecimal format.

1. **(a) Celsius to Fahrenheit Conversion** – The program converts temperature from Celsius scale© to temperature in Fahrenheit scale (F).

   Filename: Celcius-Farenheit.txt

LABEL	**Instruction**	**Operand**
TEST	START	2000
FIRST	LDA	C
	MUL	NINE
	DIV	FIVE
	ADD	THIRTYTWO
	STA	F
C	RESW	1
F	RESW	1
NINE	WORD	9
FIVE	WORD	5

THIRYTWO	WORD	32
	END	FIRST

(b) **Filename: Celcius-Farenheit_error.txt**

LABEL	**Instruction**	**Operand**
TEST	START	1003
FIRST	LDA	C
	MUL	NINE
	DIV	FIVE
	ADD	THIRTYTWO
	STA	F
C	RESW	1
F	RESW	1
FIVE	WORD	5
THIRYTWO	WORD	32
	END	FIRST

2. **(a) Simple Interest Calculation -** The following program calculates interest from the principal amount(P), rate of interest(R) and time(T).

Filename: Simple-Interest.txt

LABEL	**Instruction**	**Operand**
TEST	START	2050
FIRST	LDA	P
	MUL	RATE
	MUL	TIME
	DIV	HUNDRED
	STA	SI
P	RESW	1

```
RATE        RESW        1
TIME        RESW        1
HUNDRED     WORD        100
SI          RESW        1
END         FIRST
```

(b) Filename: Simple-Interest_error.txt

```
TEST        START       1003
FIRST       LDA         P
            MULT        RATE
            MUL         TIME
            DIV         HUNDRED
            STA         SI
P           RESW        1
RATE        RESW        1
TIME        RESW        1
HUNDRED     WORD        100
SI          RESW        1
END         FIRST
```

Opcode Table

Mnemonic	**Opcode**	**Effect**
LDA m	00	A ← (m..m + 2)
MUL m	20	A ← (A)*(m..m + 2)
DIV m	24	A ← (A)/(m..m + 2)
STA m	0C	m..m + 2 ← (A)
ADD m	18	A ← (A)+(m..m + 2)