**SYSTEMS PROGRAMMING ASSIGNMENT**

(GROUP 1)

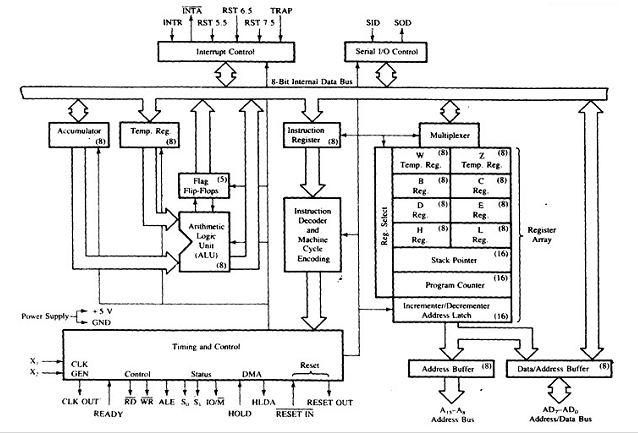
**PROBLEM STATEMENT:**

Design an 8085 assembler/simulator using C language following the working principle of Two pass assembler.

**8085 ARCHITECTURE**

8085 is an 8-bit microprocessor, designed by Intel in1977. It has the following configuration:

* 8 bit data bus
* 16 bit address bus, which can address upto 64KB
* A 16 bit program counter
* A 16 bit stack pointer
* Six 8 bit registers arranged in pairs: BC, DE, HL
* An 8 bit accumulator to perform Load/Store, I/O and ALU operations.
* An 8 bit register having 5 flags S, Z, AC, P,CY



**TWO-PASS ASSEMBLER**

An assembler is a translator, that translates an assembler program into a conventional machine language program. Basically, the assembler goes through the program one line at a time, and generates machine code for that instruction. Now, why do we need a 2-pass assembler?

Consider the following code snippet

JMP LATER

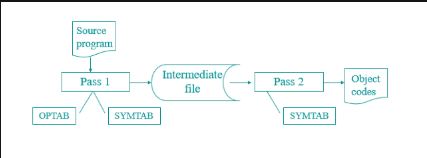
...

...

LATER:

Here assembler cannot recognise the label ‘LATER’ in the first pass and assign its address accordingly. This is known as Forward Reference problem. To address this issue, we need a 2-pass assembler.

Below is a schematic diagram off a 2-pass assembler



**DESIGN DETAILS**

The project directory contains 4 sub-directories as follows:

* **src-** Contains all the source files
* **inc-** Contains all the header files
* **data-** Contains all the data files like input file, optab, symtab
* **output-** Contains the output file generated after 2 passes

The source files are as follows:

* **hash\_map\_1.c-** Contains hashmap implementation of OPTAB.
* **hash\_map\_2.c-** Contains hashmap implementation of SYMTAB
* **to\_hex.c-** Contains utility function to convert decimal to hexadecimal
* **utils.c-** Contains utility functions used by pass 1 and pass2 of assembler
* **pass1.c-** Pass1 implementation, it takes input from data/input.txt, refers to data/optable.txt to update the locctr and updates the SYMTAB in data/symtable.txt
* **pass2.c-** Pass2 implementation, it takes input from data/input.txt, refers to data/optable.txt and data/symtable.txt to produce the output file output/output.txt. The output file is not produced in case of errors.
* **2-pass-assembler.c-** Runs the pass1 and pass2 assembler sequentially.
* **simulator.c-** Simulates the 8085 microprocessor
* **run\_program.c –** Contains utility function to run the program in 8085 simulator.

**RUNNING THE ASSEMBLER/SIMULATOR**

For compilation:

**gcc hash\_map\_1.c hash\_map\_2.c to\_hex.c utils.c pass1.c pass2.c 2-pass-assembler.c -o assembler**

**gcc run\_program.c simulator.c –o simulator**

To run the assembler on Windows, go to the Assembler/src and type **assembler.exe** on command prompt. Similarly to run the simulator go to Assembler/src and type **simulator.exe** on command prompt.

**OUTPUT**

The following is a screenshot of the input file, the symtab generated and the output object file.

