**PRACTICAL – 13**

|  |  |
| --- | --- |
|  | **Problem statement**  **Write a Program for first pass and second pass Assembler** |

**SOLUTION:**

|  |
| --- |
| **CODE:**  *// Write a Program for first pass and second pass Assembler.*  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_LABEL\_LEN 20  #define MAX\_OPCODE\_LEN 10  #define MAX\_OPERAND\_LEN 20  #define MAX\_LINE\_LEN 100  #define MAX\_SYMBOLS 100  typedef struct {      char label[MAX\_LABEL\_LEN];      int address;  } Symbol;  Symbol symbolTable[MAX\_SYMBOLS];  int symbolCount = 0;  int searchSymbol(char \**label*) {      for (int i = 0; i < symbolCount; i++) {          if (strcmp(symbolTable[i].label, *label*) == 0) {              return symbolTable[i].address;          }      }      return -1;  }  void addSymbol(char \**label*, int *address*) {      strcpy(symbolTable[symbolCount].label, *label*);      symbolTable[symbolCount].address = *address*;      symbolCount++;  }  void firstPass(FILE \**input*) {      char line[MAX\_LINE\_LEN];      int address = 0;      while (fgets(line, sizeof(line), *input*)) {          char label[MAX\_LABEL\_LEN], opcode[MAX\_OPCODE\_LEN], operand[MAX\_OPERAND\_LEN];          int numFields = sscanf(line, "%s %s %s", label, opcode, operand);          if (numFields == 3) {              if (searchSymbol(label) == -1) {                  addSymbol(label, address);              }          }          address += 4; *// Assuming each instruction is 4 bytes*      }  }  void secondPass(FILE \**input*, FILE \**output*) {      char line[MAX\_LINE\_LEN];      int address = 0;      rewind(*input*);      while (fgets(line, sizeof(line), *input*)) {          char label[MAX\_LABEL\_LEN], opcode[MAX\_OPCODE\_LEN], operand[MAX\_OPERAND\_LEN];          int numFields = sscanf(line, "%s %s %s", label, opcode, operand);          if (numFields == 3) {              fprintf(*output*, "%04X %s %s %s\n", address, label, opcode, operand);          } else if (numFields == 2) {              fprintf(*output*, "%04X %s %s\n", address, label, opcode);          } else if (numFields == 1) {              fprintf(*output*, "%04X %s\n", address, label);          }          address += 4; *// Assuming each instruction is 4 bytes*      }  }  int main() {      FILE \*input = fopen("input.asm", "r");      FILE \*output = fopen("output.obj", "w");      if (!input || !output) {          perror("Error opening file");          return EXIT\_FAILURE;      }      firstPass(input);      secondPass(input, output);      fclose(input);      fclose(output);      return EXIT\_SUCCESS;  }  **Input.asm :-**  **section .data**  **hello db 'Hello, World!', 0**  **section .text**  **global \_start**  **\_start:**  **; Write the string to stdout**  **mov eax, 4 ; syscall number for sys\_write**  **mov ebx, 1 ; file descriptor 1 is stdout**  **mov ecx, hello ; pointer to the string**  **mov edx, 13 ; length of the string**  **int 0x80 ; call kernel**  **; Exit the program**  **mov eax, 1 ; syscall number for sys\_exit**  **xor ebx, ebx ; exit code 0**  **int 0x80 ; call kernel**  **OUTPUT:** |