```c
#include <stdio.h>
#include <string.h>

int PC = 0;
int main(void) {
  FILE* input = fopen("mips.txt", "r");

  char line[50];
  char* parts[4];

  while(fgets(line, 50, input)) {
    printf("0x%08x: ", PC);
    printf("%s", line);

    split(line, parts);
    translate(parts);
    PC +=4;

  }
}
split(char* line, char** parts) {
  char* temp = strtok(line, " ");

  int i = 0;
  while(temp != NULL) {
    parts[i] = temp;
    temp = strtok(NULL, " ");
    i++;
  }
}

translate(char **parts) {
  //parts[0] = function, parts[1] = dest, parts[2] = first src, parts[3] = sec
src/imm
  int opcode;
  int rd;
  int rs;
  int rt;
  int shamt;
  int funct;
  int immediate;
  int isBranch = 0;

  //I-format
  if(strcmp(parts[0], "addi") == 0) {
    opcode = 8;
    rd = 8 + atoi(&parts[1][2]);
    rs = 8 + atoi(&parts[2][2]);
    immediate = atoi(parts[3]);
    printIFormat(opcode, rs, rd, immediate, isBranch);
  }
  else if(strcmp(parts[0], "andi") == 0) {
    opcode = 12;
    rs = 8 + atoi(&parts[2][2]);
    rd = 8 + atoi(&parts[1][2]);
    immediate = atoi(parts[3]);
    printIFormat(opcode, rs, rd, immediate, isBranch);
  }
  else if(strcmp(parts[0], "bne") == 0) {
    isBranch = 1;
    opcode = 5;
    rs = 8 + atoi(&parts[1][2]);
    rd = 8 + atoi(&parts[2][2]);
    immediate = atoi(parts[3]);
    printIFormat(opcode, rs, rd, immediate, isBranch);
  }
  //R-Format
  else if(strcmp(parts[0], "add") == 0) {
    opcode = 0;
    rs = 8 + atoi(&parts[1][2]);
    rt = 8 + atoi(&parts[2][2]);
    rd = 8 + atoi(&parts[3][2]);
    shamt = 0;
    funct = 32;

    printRFormat(opcode, rs, rt, rd, shamt, funct);
  }
  else if(strcmp(parts[0], "sub") == 0) {
    opcode = 0;
    rs = 8 + atoi(&parts[1][2]);
    rt = 8 + atoi(&parts[2][2]);
    rd = 8 + atoi(&parts[3][2]);
    shamt = 0;
    funct = 34;

    printRFormat(opcode, rs, rt, rd, shamt, funct);
  }
  else if(strcmp(parts[0], "sll") == 0) {
    opcode = 0;
    rs = 8 + atoi(&parts[1][2]);
    rt = 0;
    rd = 8 + atoi(&parts[2][2]);
    shamt = atoi(parts[3]);
    funct = 0;

    printRFormat(opcode, rs, rt, rd, shamt, funct);
  }
}

printIFormat(int opcode, int rs, int rd, int immediate, int isBranch) {
  int forBranch = immediate;

  char* opcodeBinary = malloc(6 * sizeof(char));
  for(int i = 5; i >= 0; i--) {
    opcodeBinary[i] = (opcode % 2) + '0';
    opcode /= 2;
  }
  printf("\t(I) %s ", opcodeBinary);

  char* rsBinary = malloc(5 * sizeof(char));
    for(int i = 4; i >= 0; i--) {
      rsBinary[i] = (rs % 2) + '0';
      rs /= 2;
    }
    printf("%s ", rsBinary);

  char* rdBinary = malloc(5 * sizeof(char));
  for(int i = 4; i >= 0; i--) {
    rdBinary[i] = (rd % 2) + '0';
    rd /= 2;
  }
  printf("%s ", rdBinary);

  char* immediateBinary = malloc(16 * sizeof(char));
  for(int i = 15; i >= 0; i--) {
    immediateBinary[i] = (immediate % 2) + '0';
    immediate /= 2;
  }

  if(isBranch == 1) {
    int newPC = PC + 4 + forBranch;
    printf("%s ", immediateBinary);
    printf("(Branch Address: 0x%08x)\n", newPC);
  } else {
     printf("%s\n", immediateBinary);
  }
}

printRFormat(int opcode, int rd, int rs, int rt, int shamt, int funct) {
  char* opcodeBinary = malloc(6 * sizeof(char));
  for(int i = 5; i >= 0; i--) {
    opcodeBinary[i] = (opcode % 2) + '0';
    opcode /= 2;
  }
  printf("\t(R) %s ", opcodeBinary);

  char* rsBinary = malloc(5 * sizeof(char));
    for(int i = 4; i >= 0; i--) {
      rsBinary[i] = (rs % 2) + '0';
      rs /= 2;
    }
    printf("%s ", rsBinary);

  char* rtBinary = malloc(5 * sizeof(char));
  for(int i = 4; i >= 0; i--) {
    rtBinary[i] = (rt % 2) + '0';
    rt /= 2;
  }
  printf("%s ", rtBinary);

  char* rdBinary = malloc(5 * sizeof(char));
  for(int i = 4; i >= 0; i--) {
    rdBinary[i] = (rd % 2) + '0';
    rd /= 2;
  }
  printf("%s ", rdBinary);

  char* shamtBinary = malloc(5 * sizeof(char));
  for(int i = 4; i >= 0; i--) {
    shamtBinary[i] = (shamt % 2) + '0';
    shamt /= 2;
  }
  printf("%s ", shamtBinary);

 char* functBinary = malloc(6 * sizeof(char));
  for(int i = 5; i >= 0; i--) {
    functBinary[i] = (funct % 2) + '0';
    funct /= 2;
  }
  printf("%s\n", functBinary);
}
```