דקומנטציה

Assembler

קוד האסמבלר מורכב מ 5 פונקציות עיקריות:
פונקציה ראשונה "read_lines" – מקבלת וקוראת את קובץ ה- asm. ובונה רשימה מקושרת כאשר ברשימה המקושרת מאוחסן – opcode name,rd,rs,rt,R_or_I,is_label,next

פונקציה שניה "change_labels_to_adrresses" – מקבלת את הרשימה המקושרת , משנה את ערכי הלייבלים ושומרת את מיקום השורה שלהם במקום.

פונקציה שלישית "word_data" – מקבלת את הרשימה המקושרת ומתוכה בונה רשימה מקושרת נוספת שמכילה בתוכה את כל ה word.

פונקציה רביעית "print_to_memin" – פונקציה זו מקבלת את 2 הרשימות המקושרות ומדפיסה את הערכים הנדרשים לתוך memin תוך התחשבות בכלל דרישות השאלה.

פונקציה חמישית "free_all_allocated_mem" – משחררת את הזכרון של 2 הרשימות המקושרות שנוצרו במהלך הריצה.

מצורף מטה הקוד כולל **הערות נוספות והסברים**:

```
main(int argc, char* argv[]) {
	FILE* asmfile = NULL;//Define the file pointer
	asmfile = fopen(argv[1], "r");//open the .asm file
	if (asmfile == NULL) exit(1);//If opening failed exit
	line* file_lines = read_lines(asmfile);//function return linked list
of lines from asm file
	fclose(asmfile);
	change_labels_to_adrresses(file_lines);// change label to adrresses
in linked list of lines
	word* word_lines = word_data(file_lines); //function return linked
list of word
	FILE* memin = fopen(argv[2], "w"); // open memin file
	if (memin == NULL)
		exit(1);
	print_to_memin(file_lines, word_lines, memin); // function print to
memin file
	fclose(memin);
	free_all_allocated_mem(file_lines, word_lines);	//function that
deletes all allocated memory
	return 0;
}
```

פונקציה ראשונה –

מבנה הרשימה המקושרת –

```
typedef struct line //define a struct to store a line from .acm file
{
	char opcode_name[MAX_LABEL + 2];//label can be 50 chars and the ":"
and "\0" takes 2 more chars
	char rd[MAX_LINE];
	char rs[MAX_LINE];
	char rt[MAX_LINE];
	char imm[MAX_LINE];
	struct line* next;
	int is_label; //number of line that label appears
	int R_or_I; // I format command or R format
} line;
```

- "read_lines"

```
// function read_line get the asm file and return linked list of lines
// function call to create_lines_list with a new line from asm file to all
asm bile
line* read_lines(FILE* asmfile) {
	char* new_line = (char*)malloc(MAX_LINE);//allocate memory for new
line
	line* all_lines = NULL;
	int line_counter = 0;//line counter for the labels
	if (new_line == NULL) { printf("error allocating memory"); exit(1);
}//chack allocation
	//The next loop reads the file line by line and sends it to "all_lines"
that creates
	//a linked list of all lines devided to parts
	while (((fgets(new_line, 500, asmfile)) != NULL) & (new_line != EOF))
{
		all_lines = create_lines_list(all_lines, new_line,
&line_counter);//send to a func. that builds the line linked list

	}
	free(new_line); // free memory of last line that not in use

	return all_lines;
}
```

פונקציה זו נעזרת בפונקציית עזר בשם "create_lines_list" אשר מקבלת את ראש הרשימה המקושרת, מונה של מספר השורה ואת השורה החדשה ומחזירה את ראש הרשימה המקושרת לאחר הוספת השורה החדשה לרשימה המקושרת.

```
// function create_lines_list get pointer of linked list ,new line , line
counter
// and retun the haead of the linked list
// the fuction build the linked list of line
line* create_lines_list(line* all_lines, char* new_line, int*
line_counter) {
line* new_link = (line*)malloc(sizeof(line));//allocate memory for new
link in list
int count = 0, i = 0;//count itterates over the line, i itterates over the
struct fields
int label = 0;//an indication if the current line is a label
line* ptr = NULL;//pointer to navigate throw the linked list
if (new_link == NULL) return NULL;//check allocation
new_link->is_label = -1;//reset the label flag
new_link->R_or_I = 0;//reset the format to R
int flag = 0;
//next loop fills the fieldes of the struct it has 5 repetitions for the 5
different fields
//name field
while ((new_line[count] != '\n') & (new_line[count] != ' ') &
(new_line[count] != ',') & (label == 0)) { //this while func build the
opcode_name
if (new_line[count] == '\t') {
count++;
continue;
}
else if (new_line[count] == ':') {//the odentifier of a label
label = 1;
```

```
new_link->is_label = *line_counter;//change the label field to the labels
address
*line_counter = *line_counter - 1;//we do not count labels as lines

}
new_link->opcode_name[i] = new_line[count];//copy the char to the right
field
i++;
count++;
}
new_link->opcode_name[i] = '\0';//end the field content



//rd field
i = 0;
count++;
while ((new_line[count] != '\n') & (new_line[count] != ',') & (label == 0)
& flag != 2) {
if (new_line[count] == '\t') {
count++;
continue;
}
if (new_line[count] == ' ') {
if (flag == 1) {
flag = 2;
continue;
}
count++;
continue;

}
new_link->rd[i] = new_line[count];
if (strcmp(new_link->opcode_name, ".word") == 0) flag = 1;

i++;
count++;
}
new_link->rd[i] = '\0';
i = 0;
count++;

//rs field

while ((new_line[count] != '\n') & (new_line[count] != ',') & (label ==
0)) {
if ((new_line[count] == '\t') | (new_line[count] == ' ')) {
count++;
continue;
}
new_link->rs[i] = new_line[count];
i++;
count++;
}
new_link->rs[i] = '\0';

//rt field
if (strcmp(new_link->opcode_name, ".word") == 0) {//checking if this is
the last line
new_link->imm[0] = '\0';
new_link->rt[0] = '\0';
}
else {
```

```
i = 0;
count++;
while ((new_line[count] != '\n') & (new_line[count] != ',') & (label ==
0)) {
if ((new_line[count] == '\t') | (new_line[count] == ' ')) {
count++;
continue;
}
new_link->rt[i] = new_line[count];
i++;
count++;
}
new_link->rt[i] = '\0';

//imm field
i = 0;
count++;
while ((new_line[count] != '\n') & (new_line[count] != ',') &
(new_line[count] != '#') & (label == 0)) {
if ((new_line[count] == '\t') | (new_line[count] == ' ')) {
count++;
continue;
}
new_link->imm[i] = new_line[count];
i++;
count++;
}
new_link->imm[i] = '\0';
}

//check the format type and change it only if it's I format
if (strcmp(new_link->rd, "$imm") == 0) {
new_link->R_or_I = 1;
*line_counter = *line_counter + 1;
}
else if (strcmp(new_link->rs, "$imm") == 0) {
new_link->R_or_I = 1;
*line_counter = *line_counter + 1;
}
else if (strcmp(new_link->rt, "$imm") == 0) {
new_link->R_or_I = 1;
*line_counter = *line_counter + 1;
}

//adding the new_link to the linked list for the first time
if (all_lines == NULL) {
all_lines = new_link;
all_lines->next = NULL;
*line_counter = *line_counter + 1;
return all_lines;
}
// adding the new_link to the linked list not for the first time
else {
ptr = all_lines;
while (all_lines->next != NULL) all_lines = all_lines->next;
all_lines->next = new_link;
new_link->next = NULL;
*line_counter = *line_counter + 1;
return ptr;
}
```

```
}
```

פונקציה שניה "change_labels_to_adrresses"

```
// function change_labels_to_adrresses get the head of the linked list of
line
// function change head->imm to the number of line of the label
void change_labels_to_adrresses(line* file_lines) {//need to finish it!
line* head = file_lines;//head of linked list
int line_number;
while (head != NULL) {
if ((head->imm[0] >= 'A' && head->imm[0] <= 'Z') || (head->imm[0] >= 'a'
&& head->imm[0] <= 'z')) {
line_number = find_label_and_change(file_lines, head->imm); // function
find_label_and_change return the number line of the label
itoa(line_number, head->imm, 10);
}
head = head->next;
}
return;

}
```

פונקציה זו נעזרת בפונקציית עזר "change_labels_to_adrresse" אשר מקבלת את ראש הרשימה המקושרת ואת שם הלייבל ומחזירה את מספר השורה המתאים של הלייבל.

```
//function get the heat of the linked list of lines and the label
//function return the line number of the label
// function find the line number that label is appears
int find_label_and_change(line* file_lines, char* imm) { /// change label
to his line number
	line* head1 = file_lines;//head of linked list
	char imm1[51];
	int how_many_word = 0;
	strcpy(imm1, imm);
	strcat(imm1, ":");
	while (head1 != NULL) {
		if (strcmp(head1->opcode_name, ".word") == 0) {
			how_many_word++;
		}
		if (strcmp(head1->opcode_name, imm1) == 0) {
			return ((head1->is_label) - how_many_word);
		}
		head1 = head1->next;
	}

}
```

פונקציה שלישית "word_data"

```
// function get the head of the linked list of lines and return the head
of the linked list of word
// function find .word in linked line and make a new linked list of .word
word* word_data(line* file_lines) {
	word* head_new_word = NULL;
	word* temp = NULL;
	line* head = file_lines;//pointer to navigate throw the linked list
	while (head != NULL) {
		if (strcmp(head->opcode_name, ".word") == 0) {
			word* new_word = (word*)malloc(sizeof(word));
			if (new_word == NULL) return NULL;
			//take care of hex numbers

			if (((head->rd[0] == '0') && (head->rd[1] == 'x')) ||
((head->rd[0] == '0') && (head->rd[1] == 'X'))) // if it start with haxa
				new_word->line_number =
convert_from_hex_to_dec(head->rd); // function convert_from_hex_to_dec
return the number that appears in decimal
			else
				new_word->line_number = atoi(head->rd);
			if (((head->rs[0] == '0') && (head->rs[1] == 'x')) ||
((head->rs[0] == '0') && (head->rs[1] == 'X')))// if it start with haxa
				new_word->value = convert_from_hex_to_dec(head-
>rs);  // function convert_from_hex_to_dec return the number that appears
in decimal
			else
				new_word->value = atoi(head->rs);
			// adding new_word to the linked list of word
			if (temp == NULL) {
				temp = new_word;
				head_new_word = temp;
				temp->next = NULL;
			}
			else {
				temp->next = new_word;
				temp = temp->next;
				temp->next = NULL;
			}
		}

		head = head->next;
	}
	return head_new_word;
}
```

פונקציה זו נעזרת בפונקציית עזר "convert_from_hex_to_dec" אשר מקבלת מספר אשר רשום בהקסה דצימלי ומחזירה מספר בערך צדימלי.

```
// function get hex char number with start of 0x and return number as int
//function convert hex number to decimal number
int convert_from_hex_to_dec(char hex[]) {
	int length = 0, decimal = 0, base = 1;
	int end = 0;
	char hex1[6];
	int len_hex = strlen(hex);
	for (int j = 0; j < len_hex - 2; j++) {
		hex1[j] = hex[j + 2];
```

```
			end = j;
		}
		hex1[end + 1] = '\0';
		length = strlen(hex1);
		for (int i = length--; i >= 0; i--)
		{
			if (hex1[i] >= '0' && hex1[i] <= '9')
			{
				decimal += (hex1[i] - 48) * base;
				base *= 16;
			}
			else if (hex1[i] >= 'A' && hex1[i] <= 'F')
			{
				decimal += (hex1[i] - 55) * base;
				base *= 16;
			}
			else if (hex1[i] >= 'a' && hex1[i] <= 'f')
			{
				decimal += (hex1[i] - 87) * base;
				base *= 16;
			}
		}
		return decimal;
}
```

פונקציה רביעית "print_to_memin" –

```
// function get linked list of lines, linked loist of word and memin file
//function print to memin as required
void print_to_memin(line* file_lines, word* word_line, FILE* memin) {
		int line_counter = 0;//in order to add zeroes at the end
		int there_is_word = 0;
		int higest_line_word = 0;
		int value_word = 0;
		line* head = file_lines;//head of the lines list
		word* head_word = word_line;
		int if_word = 0;
		while (head != NULL) {
			if (head->is_label == -1) {   //not label
				there_is_word = checks_if_exists(word_line,
line_counter); //check if word existes in current line (1 = if exists)
				if (there_is_word == 1) { //if exists =1
					value_word = find_value_word(word_line,
line_counter); // find value for relevante word
					fprintf(memin, "%05X\n", value_word);
					line_counter++;
				}
				else {
					if_word = convert_opcode_to_hexa(head-
>opcode_name, memin); //check if not word function return 0 if opcode name
is word else return 1
					if (if_word != 0) {
						print_var(head->rd, memin); // print rd to
memin
						print_var(head->rs, memin); // print rs to
memin
						print_var(head->rt, memin); // print rt to
memin
						fprintf(memin, "\n");
						line_counter = line_counter + 1;
					}
				}
				if (head->R_or_I == 1) {// i format. there is imm
```

```
                        there_is_word = checks_if_exists(word_line,
line_counter);//check if word existes in current line (1 = if exists)
                        if (there_is_word == 1) { // if word
                              value_word = find_value_word(word_line,
line_counter);// find relvante value for word
                              fprintf(memin, "%05X", value_word);
                        }
                        else {
                              printimm(head->imm, memin); // print imm
to memin with sign extension
                        }
                        fprintf(memin, "\n");
                        line_counter = line_counter + 1;
                  }
            }
            head = head->next;

      }
      higest_line_word = find_higest_line_word(word_line, line_counter);
//if left lines between the end of ths file till word line
      for (int i = line_counter; i <= higest_line_word; i++) {
            there_is_word = checks_if_exists(word_line, i); //check if
word existes in current line (1 = if exists)
            if (there_is_word == 1) {
                  value_word = find_value_word(word_line, i); // find
relvante value for word
                  if (value_word >= 0)
                        fprintf(memin, "%05X\n", value_word);
                  else {
                        value_word = abs(value_word);
                        value_word = (~(value_word - 1));
                        fprintf(memin, "%05X", value_word & 0xfffff);
                  }
            }
            else fprintf(memin, "00000\n");
      }
                                                                        }
```

כאשר פונקציה זו נעזרת בכמה פונקציות עזר :

1. **Checks_if_exists**
2. **Find_value_word**
3. **Convert_opcode_to_hexa**
4. **Print_var**
5. **Printimm**
6. **Find_higest_line_word**

1. **checks_if_exists מקבלת את ראש הרשימה המקושרת של word. ומספר שורה ומחזירה 1 אם word. אמורה להכתב שם ו 0 אחרת.**

```
//function get linked list of word and current line
//return 1 if there word should be wriiten to current line else 0
//function check if there is a word that should be written in current line
int checks_if_exists(word* word_line, int line_counter) {
	int exists = 0;
	word* head_word = word_line;
	while (head_word != NULL && exists == 0) {
		if (head_word->line_number == line_counter) {
			exists = 1;
		}
		head_word = head_word->next;
	}
	return exists;
}
```

2. find_value_word מקבלת את ראש הרשימה המקושרת של word. ומספר שורה ומחזירה את הערך אשר אמור להרשם בשורה זו.

```
//function get linked list of word and current line
//return value of word that should be written
//function find relevant value of the relevant word considre by current
line
int find_value_word(word* word_line, int line_counter) {
int value = 0;
word* head_word = word_line;
while (head_word != NULL && value == 0) {
if (head_word->line_number == line_counter) {
value = head_word->value;
}
head_word = head_word->next;
}
return value;
}
```

3. **convert_opcode_to_hexa - מקבלת את ה opcode_name ואת הקובץ memin ורושמת לתוכו את מספר ההקסה המתאים לopcode_name.**

```
//function get opcode name and memin file and retur the value of the
opcode name
// function check what value is suitable for opcode name and then print it
to memin
// if opcode name is .word retuen 0 else 1
int convert_opcode_to_hexa(char* opcode_name, FILE* memin) { // covert
opcode to hexa and print
	if (strcmp(opcode_name, "add") == 0) {
		fprintf(memin, "00"); return 1;
	}
	else if (strcmp(opcode_name, "sub") == 0) {
		fprintf(memin, "01"); return 1;
	}
	else if (strcmp(opcode_name, "mul") == 0) {
		fprintf(memin, "02"); return 1;
	}
```

```
else if (strcmp(opcode_name, "and") == 0) {
      fprintf(memin, "03"); return 1;
}
else if (strcmp(opcode_name, "or") == 0) {
      fprintf(memin, "04"); return 1;
}
else if (strcmp(opcode_name, "xor") == 0) {
      fprintf(memin, "05"); return 1;
}
else if (strcmp(opcode_name, "sll") == 0) {
      fprintf(memin, "06"); return 1;
}
else if (strcmp(opcode_name, "sra") == 0) {
      fprintf(memin, "07"); return 1;
}
else if (strcmp(opcode_name, "srl") == 0) {
      fprintf(memin, "08"); return 1;
}
else if (strcmp(opcode_name, "beq") == 0) {
      fprintf(memin, "09"); return 1;
}
else if (strcmp(opcode_name, "bne") == 0) {
      fprintf(memin, "0A"); return 1;
}
else if (strcmp(opcode_name, "blt") == 0) {
      fprintf(memin, "0B"); return 1;
}
else if (strcmp(opcode_name, "bgt") == 0) {
      fprintf(memin, "0C"); return 1;
}
else if (strcmp(opcode_name, "ble") == 0) {
      fprintf(memin, "0D"); return 1;
}
else if (strcmp(opcode_name, "bge") == 0) {
      fprintf(memin, "0E"); return 1;
}
else if (strcmp(opcode_name, "jal") == 0) {
      fprintf(memin, "0F"); return 1;
}
else if (strcmp(opcode_name, "lw") == 0) {
      fprintf(memin, "10"); return 1;
}
else if (strcmp(opcode_name, "sw") == 0) {
      fprintf(memin, "11"); return 1;
}
else if (strcmp(opcode_name, "reti") == 0) {
      fprintf(memin, "12"); return 1;
}
else if (strcmp(opcode_name, "in") == 0) {
      fprintf(memin, "13"); return 1;
}
else if (strcmp(opcode_name, "out") == 0) {
      fprintf(memin, "14"); return 1;
}
else if (strcmp(opcode_name, "halt") == 0) {
      fprintf(memin, "15"); return 1;
}
return 0;
                                                                    }
```

4. print_var - מקבלת רגיסטר rs,rd,rt ואת הקובץ memin ורושמת לתוכו את הרגיסטר בהתאמה בהקסהצדימלי.

```
// function get variable (rd,rs,rt) and memin file
// function check what value is suitable for variable and then print it to
memin
void print_var(char* var, FILE* memin) { // convert rt,rs,rd to hexa
	if (strcmp(var, "$zero") == 0)
		fprintf(memin, "0");
	else if (strcmp(var, "$imm") == 0)
		fprintf(memin, "1");
	else if (strcmp(var, "$v0") == 0)
		fprintf(memin, "2");
	else if (strcmp(var, "$a0") == 0)
		fprintf(memin, "3");
	else if (strcmp(var, "$a1") == 0)
		fprintf(memin, "4");
	else if (strcmp(var, "$a2") == 0)
		fprintf(memin, "5");
	else if (strcmp(var, "$a3") == 0)
		fprintf(memin, "6");
	else if (strcmp(var, "$t0") == 0)
		fprintf(memin, "7");
	else if (strcmp(var, "$t1") == 0)
		fprintf(memin, "8");
	else if (strcmp(var, "$t2") == 0)
		fprintf(memin, "9");
	else if (strcmp(var, "$s0") == 0)
		fprintf(memin, "A");
	else if (strcmp(var, "$s1") == 0)
		fprintf(memin, "B");
	else if (strcmp(var, "$s2") == 0)
		fprintf(memin, "C");
	else if (strcmp(var, "$gp") == 0)
		fprintf(memin, "D");
	else if (strcmp(var, "$sp") == 0)
		fprintf(memin, "E");
	else if (strcmp(var, "$ra") == 0)
		fprintf(memin, "F");
}
```

5. printimm – מקבלת רגיסטר imm ואת הקובץ memin ורושמת לתוך הקובץ memin את תוכן הרגיסטר imm כנדרש.

```
//function get imm and memin file
// function print imm to memin file and make sign extension
void printimm(char* imm, FILE* memin) {
	if ((imm[0] == '0' && imm[1] == 'x') | (imm[0] == '0' && imm[1] ==
'X')) { // Checks if exhadmili number
		char hex_imm[52];//uses to store the hex input
		change_to_right_format_and_print(hex_imm, imm, memin); //
function print to memin if imm is hex
	}
	else {
		int num = atoi(imm);
		if (num >= 0) //Checks if a positive number
			fprintf(memin, "%05X", num);
		if (num < 0) { // Checks for a negative number
			num = abs(num);
			num = (~(num - 1));
			fprintf(memin, "%05X", num & 0xfffff);
		}
	}

}
```

כאשר גם פונקציה זו נעזרת בפונקצית עזר בשם "change_to_right_format_and_print" במידה וקיבלנו את הערך בספרות הקסה דצימליות אז נדפיס בהתאם לקובץ memin .

הפונקציה מקבלת את המספר בייצוג הקסה דצמילי, ואת הקובץ memin ומדפיסה לתוכו את המספר על פי הדרישה.

```
//function get hex_imm, string imm anf memin file
// function print imm to memin in case imm is hex
void change_to_right_format_and_print(char* hex_imm, char* imm, FILE* memin)
{
	int len = strlen(imm);
	int j = 0;
	for (int i = 2; i < len; i++) {
		hex_imm[i - 2] = imm[i];//copy all he chars after the second
char
		j = i;
	}
	hex_imm[j - 1] = '\0';//finish the string
	len = strlen(hex_imm);
	if (len == 5)
		fprintf(memin, "%s", hex_imm);
	else if (len == 4)
		fprintf(memin, "0%s", hex_imm);
	else if (len == 3)
		fprintf(memin, "00%s", hex_imm);
	else if (len == 2)
		fprintf(memin, "000%s", hex_imm);
	else if (len == 1)
		fprintf(memin, "0000%s", hex_imm);
}
```

6. find_higest_line_word – מקבלת את מספר השורה ובנוסף את ראש הרשימה המקושרת של word. ומחזירה את את מספר השורה הגדול ביותר אשר word. רושמת לתוכו.

```
//function get linked list of word and current line
//return the higest line number that word exists
// function find the higest line number that word exists
int find_higest_line_word(word* word_line, int line_counter) {
	int higest_value_word = 0;
	word* head_word = word_line;
	while (head_word != NULL) {
		if (head_word->line_number > higest_value_word) {
			higest_value_word = head_word->line_number;
		}
		head_word = head_word->next;
	}
	return higest_value_word;
}
```

פונקציה חמישית "free_all_allocated_mem"

```
//function fet linked list of lines and linked list of word
// free all memory from both linked list
void free_all_allocated_mem(line* all_lines, word* all_words) {
	line* curr_line = NULL;
	word* curr_word = NULL;
	while (all_lines != NULL) {
		curr_line = all_lines;
		all_lines = all_lines->next;
		free(curr_line);
	}
	free(all_lines);
	while (all_words != NULL) {
		curr_word = all_words;
		all_words = all_words->next;
		free(curr_word);
	}
	free(all_words);

	return;
}
```

Simulator

קוד הסימולטור מורכב מ 8 פונקציות כאשר העיקרית מבינהם היא "print_to_trace" אשר בה מתבצע תהליך הפסיקות וביצוע סט ההוראות ומפונקציה זאת נעזרים על מנת לייצא את רוב הקבצים הנדרשים.

בפונקצית ה – "main" ישנם את הפונקציות הבאות:

1. Diskin_to_array
2. Memin_to_array
3. Print_to_trace
4. Print_to_monitor
5. Print_to_diskout
6. Print_to_regout
7. Print_to_cycles
8. Print_to_memout

מתוך פונקציות אלה פונקציות 1 ו-2 הן לקריאת הנתונים מקבצי הקלט diskin , memin . פונקציה 3 print_to_trace בה מתבצעים חלק משמעותי מן הפעולות למימוש הלוגי של הסימולטור והפונקציות הנותרות נעזרות במידע אשר בוצע בפונקצית print_to_trace על מנת לבצע את ייעודן ולהדפיס את המידע הרלוונטי לקבצים של monitor, diskout,regout,cycle,memout.

```
//main - gets argv, argc as inputs
//main function uses to open and close files and send to other important
functions
//sets all the needed arrays to store memory and registers

int main(int argc, char* argv[]) {
	FILE* memin = fopen(argv[1], "r");
	int memin_array[4097] = { 0 };
	int disk[128 * 128] = { 0 };
	int pc = 0;//pc counter
	int registers[16] = { 0 };
	int io_registers[23] = { 0 };
	int cycles = 0;//cycle counter
	int monitor_array[256 * 256] = { 0 };

	FILE* diskin = fopen(argv[2], "r");
	if (diskin == NULL) {//check if file opening worked
		printf("Error Opening diskin.txt");
		exit(1);
	}

	diskin_to_array(diskin, disk);


	FILE* irq2 = fopen(argv[3], "r");
	if (irq2 == NULL) {//check if file opening worked
		printf("Error Opening irq2.txt");
		exit(1);
	}


	FILE* monitor = fopen(argv[12], "w");
	if (monitor == NULL) {
		printf("Error opening monitor file");
		exit(1);
	}
	FILE* monitor_yuv = fopen(argv[13], "w");
	if (monitor_yuv == NULL) {
		printf("Error opening monitor.yuv file");
```

```
		exit(1);
	}

	FILE* memin_file = fopen(argv[1], "r");
	if (memin_file == NULL) { printf("error opening file"); exit(1); }
	memin_to_array(memin_array, memin_file);
	fclose(memin_file);

	FILE* trace = fopen(argv[6], "w");
	if (trace == NULL) {
		printf("Error opening trace file");
		exit(1);
	}
	FILE* hwregtrace = fopen(argv[7], "w");
	if (hwregtrace == NULL) {
		printf("Error opening hwregtrace file");
		exit(1);
	}
	FILE* leds = fopen(argv[9], "w");
	if (leds == NULL) {
		printf("Error opening leds file");
		exit(1);
	}
	FILE* display7seg = fopen(argv[10], "w");
	if (leds == NULL) {
		printf("Error opening display7seg file");
		exit(1);
	}
	FILE* diskout = fopen(argv[11], "w");
	if (leds == NULL) {
		printf("Error opening diskout file");
		exit(1);
	}

	print_to_trace(pc, trace, registers, io_registers, memin_array,
&cycles, irq2, hwregtrace, leds, display7seg, monitor_array, disk);//write
to trace.txt and execute commands
	print_to_monitor(monitor_array, monitor, monitor_yuv);
	print_to_diskout(diskout, disk);
	fclose(trace);
	fclose(irq2);//the file is used in print to trace and thus closed
only here
	fclose(hwregtrace);
	fclose(leds);
	fclose(display7seg);
	fclose(monitor);
	fclose(monitor_yuv);
	fclose(diskout);
	FILE* regout = fopen(argv[5], "w");
	if (regout == NULL) { printf("error opening a file"); exit(1); }
	print_to_regout(regout, registers);//write to regout.txt the content
of registers
	fclose(regout);

	FILE* cycles_file = fopen(argv[8], "w");
	if (cycles_file == NULL) {
		printf("Error opening cycles file");
		exit(1);
	}
	print_to_cycles(cycles_file, cycles);
	fclose(cycles_file);

	FILE* memout = fopen(argv[4], "w");
```

```
	if (memout == NULL) {
		printf("Error opening memout file");
		exit(1);
	}
	print_to_memout(memout, memin_array);
	fclose(memout);
	return 0;
}
```

<u>פונקציה ראשונה "diskin_to_array" -</u> מקבלת את הקובץ diskin ומערך disk, הפונקציה מכניסה את הערכים מהקובץ אל תוך המערך תוך התחשבות במשלים ל2.

```
//next function gets diskin array and file, and copies diskin file to an
int array
void diskin_to_array(FILE* diskin, int* disk) {
	char line[10] = "line";
	int i = 0;
	int check = 0;
	int test = 0;
	while (fgets(line, 7, diskin) != NULL) {
		check = sscanf(line, "%X", &disk[i]);
		if (line[0] == '8' || line[0] == '9' || line[0] == 'A' ||
line[0] == 'a' ||
			line[0] == 'B' || line[0] == 'b' ||
			line[0] == 'C' || line[0] == 'c' ||
			line[0] == 'D' || line[0] == 'd' ||
			line[0] == 'E' || line[0] == 'e' ||
			line[0] == 'F' || line[0] == 'f') {
			test = disk[i];
			test = test | 0xFFF00000;
			disk[i] = test;
		}

		i++;
	}


	return;
}
```

<u>פונקציה שניה "memin_to_array" –</u> מקבלת בקלט את קובץ memin ואת המערך memin , הפונקציה מכניסה את הערכים מתוך קובץ memin לתוך מערך memin.

```
// memin_to_memarray gets the array of memin and the memin file
//and copies the files content to the memin (int) array
void memin_to_array(int memin[4096], FILE* memin_file) {
	char line[10] = "line";
	int test = 0;
	int i = 0;
	while (fgets(line, 7, memin_file) != NULL) {
		if (line[0] == '8' || line[0] == '9' || line[0] == 'A' ||
line[0] == 'a' ||
			line[0] == 'B' || line[0] == 'b' ||
			line[0] == 'C' || line[0] == 'c' ||
			line[0] == 'D' || line[0] == 'd' ||
			line[0] == 'E' || line[0] == 'e' ||
			line[0] == 'F' || line[0] == 'f') {
			test = memin[i];
			test = test | 0xFFF00000;
```

```
                memin[i] = test;
        }

        i++;
    }

                                                                    }
```

<u>פונקציה שלישית "print_to_trace"</u> – מקבלת את ערך ה pc ההתחלתי, קובץ trace, מערך registers, מערך io_registers, מערך memin, קובץ cycles, קובץ irq2_file, קובץ hwregtrace, קובץ leds, קובץ display7seg, מערך monitor_array ומערך disk כאשר בפונקציה זו מתבצעת התחשבות בפסיקות וסט ההוראות בעזרת פונקצית עזר "change_registers_after_print".

```
//next function prints the currnt content of all registers to trace file
//updates imm register if needed
//and then sends them to a function that updates the registers and the pc
void print_to_trace(int pc, FILE* trace, int* registers, int*
io_registers, int* memin, int* cycles, FILE* irq2_file, FILE* hwregtrace,
FILE* leds, FILE* display7seg, int* monitor_array, int* disk) {
    int counter = 0;//for test
    int irq = 0;//an int to identify interrupts
    int cycle_counter = 0;
    char* line[MAX_LINE];
    fgets(line, MAX_LINE, irq2_file);//read first line from irq2 file
    int irq2 = atoi(line);//if we have an interrupt from irq2 this int
will contain cycle number
    while (pc != -1) {
        int r_or_i = 0;//I type or R type command
        int op_code = 0;
        int rd = 0, rt = 0, rs = 0;
        if (io_registers[5] != 1) {
            if (irq2 == *cycles - 1 || irq2 == *cycles - 2 || irq2
== *cycles - 3) {
                io_registers[5] = 1;//set status to 1 -stil
handeling interrupt!
                fgets(line, MAX_LINE, irq2_file);
                irq2 = atoi(line);
            }
            if ((io_registers[0] & io_registers[3]) |
(io_registers[1] & io_registers[4]) | (io_registers[2] & io_registers[5]))
{
                io_registers[7] = pc;
                pc = io_registers[6];//go to the address in
irqhandler
                continue;
            }
        }

        //print PC to trace
        fprintf(trace, "%03X ", pc);
        //print INST to trace
        fprintf(trace, "%05X ", memin[pc]);
        if (pc == 128)
            printf("%05X", memin[pc]);
        op_code = ((memin[pc] & 0x000ff000) >> 12);
        rd = ((memin[pc] & 0x00000f00) >> 8);
        rs = ((memin[pc] & 0x000000f0) >> 4);
        rt = ((memin[pc] & 0x0000000f));

        //testing if we have an I type format
        //and updating io registers (timer and clks)
```

```
            if ((op_code >= 0 && op_code <= 8) || op_code == 16 ||
op_code == 19) {
                  if (rs == 1 || rt == 1) {
                        registers[1] = memin[pc + 1];
                        *cycles = *cycles + 1;
                        if (io_registers[17] == 1 && cycle_counter !=
1024 && io_registers[14] != 0)
                              cycle_counter = cycle_counter + 1;
                        io_registers[8] += 1;
                        if (io_registers[8] >= 0xffffffff)
                              io_registers[8] = 0;
                        if (io_registers[11] == 1) {
                              if (io_registers[12] < io_registers[13]) {
                                    io_registers[12]++;
                              }
                              else {
                                    io_registers[12] = 0;
                                    io_registers[3] = 1;
                              }
                        }
                  }
                  else
                        registers[1] = 0;
            }
            else if ((op_code >= 9 && op_code <= 14) || op_code == 17 ||
op_code == 20) {
                  if (rd == 1 || rs == 1 || rt == 1) {
                        registers[1] = memin[pc + 1];
                        *cycles = *cycles + 1;
                        if (io_registers[17] == 1 && cycle_counter !=
1024 && io_registers[14] != 0)
                              cycle_counter = cycle_counter + 1;
                        io_registers[8] += 1;
                        if (io_registers[8] >= 0xffffffff)
                              io_registers[8] = 0;
                        if (io_registers[11] == 1) {
                              if (io_registers[12] < io_registers[13]) {
                                    io_registers[12]++;
                              }
                              else {
                                    io_registers[12] = 0;
                                    io_registers[3] = 1;
                              }
                        }
                  }
                  else
                        registers[1] = 0;
            }
            else if (op_code == 15) {
                  if (rs == 1) {
                        registers[1] = memin[pc + 1];
                        *cycles = *cycles + 1;
                        if (io_registers[17] == 1 && cycle_counter !=
1024 && io_registers[14] != 0)
                              cycle_counter = cycle_counter + 1;
                        io_registers[8] += 1;
                        if (io_registers[8] >= 0xffffffff)
                              io_registers[8] = 0;
                        if (io_registers[11] == 1) {
                              if (io_registers[12] < io_registers[13]) {
                                    io_registers[12]++;
                              }
                              else {
```

```
                                    io_registers[12] = 0;
                                    io_registers[3] = 1;
                              }
                        }
                  }
                  else
                        registers[1] = 0;
            }
            else
                  registers[1] = 0;


            //print all 16 registers to trace- before making the change
            for (int i = 0; i < 15; i++) {
                  fprintf(trace, "%08X ", registers[i]);
            }
            fprintf(trace, "%08X", registers[15]);
            fprintf(trace, "\n");

            //next line sends the registers to be changed according to
the command
            //and returns the next PC

            pc = change_registers_after_print(registers, pc,
io_registers, memin[pc], memin, cycles, hwregtrace, leds, display7seg,
monitor_array, disk, &cycle_counter);
            counter++;

      }
}
```

פונקצית "print_to_hwregtrace_leds_display7seg_and_update_monitor" – מקבלת כקלט את מספר הcycle, int read_write אשר נותן מידע האם יש פעולת קריאה או כתיבה, reg_number מספר ריגסטור החומרה, content אשר מכיל בטוחו את המידע , קובץ hwregtrace, קובץ leds, קובץ display7seg, מערך monitor_array ומערך של io_registers. פונקציה זו מתעסקת עם רגיסטרי החומרה וכותבת לתוך הקבצים hwregout,leds,display7seg כאשר מתבצעת פעולה מתאימה.

```
//next function is called from change_registers_after_print
//it is used to print to hwregtrace file
//the inputs are - current number of cycles, read or write int - 0=read,
1=write
//the io register number, the data , file pointers: hwregtrace  leds and
display7seg
void print_to_hwregtrace_leds_display7seg_and_update_monitor(int* cycles,
int read_write, int reg_number, int content, FILE* hwregtrace, FILE* leds,
FILE* display7seg, int* monitor_array, int* io_registers) {
      fprintf(hwregtrace, "%d ", cycles);
      if (read_write == 0)
            fprintf(hwregtrace, "READ ");
      else
            fprintf(hwregtrace, "WRITE ");
      if (reg_number == 0)
            fprintf(hwregtrace, "irq0enable ");
      else if (reg_number == 1)
            fprintf(hwregtrace, "irq1enable ");
      else if (reg_number == 2)
            fprintf(hwregtrace, "irq2enable ");
      else if (reg_number == 3)
            fprintf(hwregtrace, "irq0status ");
```

```
	else if (reg_number == 4)
		fprintf(hwregtrace, "irq1status ");
	else if (reg_number == 5)
		fprintf(hwregtrace, "irq2status ");
	else if (reg_number == 6)
		fprintf(hwregtrace, "irqhandler ");
	else if (reg_number == 7)
		fprintf(hwregtrace, "irqreturn ");
	else if (reg_number == 8)
		fprintf(hwregtrace, "clks ");
	else if (reg_number == 9) {
		fprintf(hwregtrace, "leds ");
		if (read_write == 1)
			fprintf(leds, "%d %08X\n", cycles, content);
	}
	else if (reg_number == 10) {
		fprintf(hwregtrace, "display7seg ");
		if (read_write == 1)
			fprintf(display7seg, "%d %08X\n", cycles, content);
	}
	else if (reg_number == 11)
		fprintf(hwregtrace, "timerenable ");
	else if (reg_number == 12)
		fprintf(hwregtrace, "timercurrent ");
	else if (reg_number == 13)
		fprintf(hwregtrace, "timermax ");
	else if (reg_number == 14)
		fprintf(hwregtrace, "diskcmd ");
	else if (reg_number == 15)
		fprintf(hwregtrace, "disksector ");
	else if (reg_number == 16)
		fprintf(hwregtrace, "diskbuffer ");
	else if (reg_number == 17)
		fprintf(hwregtrace, "diskstatus ");
	else if (reg_number == 18 || reg_number == 19)
		fprintf(hwregtrace, "reserved ");
	else if (reg_number == 20)
		fprintf(hwregtrace, "monitoraddr ");
	else if (reg_number == 21)
		fprintf(hwregtrace, "monitordata ");
	else if (reg_number == 22) {
		fprintf(hwregtrace, "monitorcmd ");
		if (read_write == 1) { // change value of pixel , check if
write
			if (io_registers[22] == 1) {
				monitor_array[io_registers[20]] =
io_registers[21]; //change value(monitordata)
				//of pixel in position (monitoradder)
			}
		}
	}
	fprintf(hwregtrace, "%08X\n", content);
	return;

}
```

<u>פונקציה רביעית "print_to_monitor" -</u> מקבלת כקלט את מערך monitor_array , קובץ monitor, קובץ monitor_yuv , פונקציה זו כותבת לקבצים monitor, וmonitor_yuv .

```
//next function prints to monitor.txt and monitor.yuv files
void print_to_monitor(int* monitor_array, FILE* monitor, FILE*
monitor_yuv) {
	for (int i = 0; i < 256 * 256; i++) {
```

```
			fprintf(monitor, "%02X\n", monitor_array[i]);
			fwrite(&monitor_array[i], 1, 1, monitor_yuv);
		}

		return;
}
```

<u>פונקציה חמישית "print_to_diskout" –</u> מקבלת קובץ diskout , ומערך של disk. הפונקציה מדפיסה לתוך diskout את המערך disk לערך שבוצעו בו שינוים בפונקצית print_to_tace.

```
//print to disk - input: diskout file pointer, disk array
//prints disk array content to diskout file
//returns void
void print_to_diskout(FILE* diskout, int* disk) {
		for (int i = 0;i < (128 * 128);i++) {
			fprintf(diskout, "%05X\n", disk[i]);
		}
}
```

<u>פונקציה שישית "print_to_regout" –</u> מקבלת כקלט את קובץ regout ואת המערך registers וכותבת לתוך regout את את המערת registers.

```
//next function prints the content of the registers R2-R15
//after printing to trace.txt the registers contain the updated info
//and therefor we just need to print them as they are
void print_to_regout(FILE* regout, int* registers) {
		for (int i = 2; i < 16; i++) {
			fprintf(regout, "%08X\n", registers[i]);
		}
		return;
}
```

<u>פונקציה שביעית "print_to_cycles" –</u> מקבלת כקלט את קובץ cycle ואת משתנה cycle ומדפיסה את cycle לתוך הקובץ.

```
//next lines print the number of cycles to a new file named cycles
void print_to_cycles(FILE* cycles, int cyc) {
		fprintf(cycles, "%d", cyc);
}
```

<u>פונקציה שמינית "print_to_memout" –</u> מקבלת כקלט את קובץ memout ואת המערך memin ומדפיהס לתוך הקובץ את מערך memin לאחר שבוצע השינוי שם סט ההוראות.

```
//next function prints to memout file
//the array memin is now updated and finalized and ready to be printed to
memout
void print_to_memout(FILE* memout, int* memin) {
		for (int i = 0; i < 4097; i++) {
			fprintf(memout, "%05X\n", (0x000fffff & memin[i]));
		}
}
```