Αναφορά 3ου Εργαστηρίου

<u>Ομάδα</u>

<u>Προεργασία</u>

Για την υλοποίηση του 3ου εργαστηρίου μας ζητήθηκε ,σε γλώσσα C και Clang, να τροποποιήσουμε την κλήση των συναρτήσεων του προηγούμενου εργαστηρίου χρησιμοποιώντας function pointers.Επιπλέον μας ζητήθηκε να υπολογίσουμε το πραγματικό μέγεθος της λίστας μας καθώς και να ενσωματώσουμε στους κώδικες μας μία νέα συνάρτηση που εκτυπώνει τις διευθύνσεις όλων των συναρτήσεων μέσα στο πρόγραμμα. Όσον αφορά τη γλώσσα Assembly ,θα πρέπει να κατασκευάσουμε το μενού επιλογής του χρήστη με όλους τους απαραίτητους βρόγχους (while loop και if).

<u>Περιγραφή Ζητούμενων</u>

Ο σκοπός της άσκησης αρχικά ,είναι να κατανοήσουμε την κλήση των συναρτήσεων στον κώδικα μέσω function pointers .Έπειτα σκοπός είναι να υλοποιήσουμε προγράμματα σε γλώσσα C και Clang , με χρήση καταχωρητών σε όλη την έκταση του κώδικα ,και όχι μεταβλητών όπου δεν είναι απαραίτητο. Μ' αυτόν τον τρόπο όλες οι λειτουργίες του προγράμματος θα πρέπει να γίνονται μέσω global καταχωρητών τύπου integer ,κάνοντας όπου είναι απαραίτητο για τη λειτουργικότητα του προγράμματος, τα κατάλληλα typecasts.Στην Assembly καλούμαστε να υλοποιήσουμε το μενού επιλογής του χρήστη ,ο οποίος θα εισάγει τον αριθμό επιλογής και θα εκτυπώνεται ο ακέραιος μαζί με το μήνυμα εισόδου της επόμενης επιλογής. Για να υλοποιηθεί αυτό θα γίνει χρήση εντολών branch για τις συνθήκες και εντολών jump για την εκτέλεση των βρόγχων.

<u>Περιγραφή Εκτέλεσης</u>

Αρχικά ενσωματώσαμε στους κώδικες C και Clang του προηγούμενου εργαστηρίου τους function pointers για την κλήση των συναρτήσεων ,όπως φαίνεται παρακάτω:

C (χωρίς function pointers)	C (με function pointers)	Clang (με function pointers)

int I;
i=menuPrint(); //κλήση
	int(foo)(void);//αρχικοποίηση
	foo=&menuPrint;
	
	foo(); //κλήση
		int R15;
		main() {….
		
		int(foo)(void);//αρχικοποίηση
		foo=&menuPrint;
		R15=(int)foo;
		jal(R15); // κλήση

Στη συνέχεια τροποποιήσαμε την συνάρτηση στην C που εκτυπώνει το μέγεθος της λίστας. Υπολογίσαμε και εκτυπώσαμε στο πραγματικό μέγεθος της λίστας μας σε byte υπολογίζοντας το πραγματικό μέγεθος του πρώτου κόμβου και πολλαπλασιάζοντάς το, με τον αριθμό των κόμβων της λίστας, όπως φαίνεται παρακάτω:

Πριν τη μετατροπή	Μετά την μετατροπή

printf("Size of list: %d\n",numOfNodes*12);	first = (int)(node); //πρώτος κόμβος
	last = (int)(node->next); //δεύτερος κόμβος
	
	printf("Size of list: %d\n",numOfNodes*(last-first));

Έπειτα ,στην γλώσσα Clang χρησιμοποιήσαμε, όπου ήταν δυνατό ,στην θέση των μεταβλητών,καταχωρητές κάνοντας τα κατάλληλα typecasts.Για παράδειγμα :

Η συνάρτηση createList

struct list * createList(){
struct list *node;
R4=(int)node;
R4 = (int)NULL;
R2 = R4;
JR31;
}

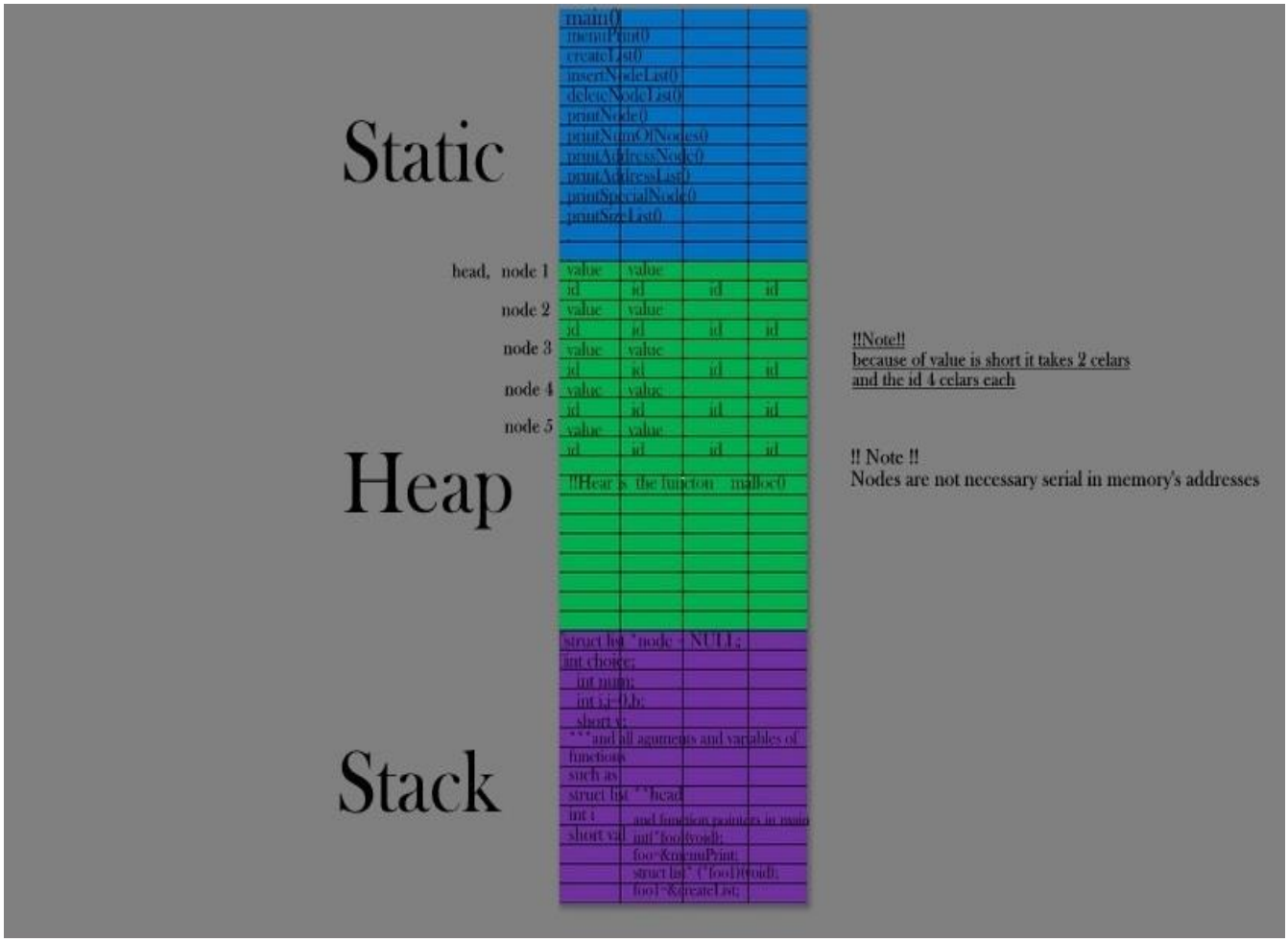
Τέλος, στον κώδικα Assembly ,κάναμε χρήση των εντολών branch και jump για να υλοποιήσουμε το loop και τα if που συμπεριλαμβάνει ο κώδικας του μενού επιλογής χρήστη όπως φαίνεται παρακάτω στο παράρτημα.

<u>Συμπεράσματα</u>

Εκτελώντας την άσκηση κατανοήσαμε την χρήση των function pointers για την κλήση των συναρτήσεων στην C και στην Clang. Επίσης, εξοικειωθήκαμε με την αποκλειστική χρήση καταχωρητών σε όλες τις συναρτήσεις του προγράμματος σε Clang ,κάνοντας typecast όπου χρειαζόταν έτσι ώστε το πρόγραμμά μας να είναι λειτουργικό. Τέλος, στην Assembly μάθαμε να χρησιμοποιούμε τις εντολές ελέγχου ροής (branch και jump) για να κατασκευάσουμε τους βρόγχους από το μενού επιλογής.

<u>Παράρτημα-Κώδικες-Χάρτης Μνήμης</u>

Πλήρης Χάρτης Μνήμης:



Κώδικας σε γλώσσα C

```
#include <stdio.h>

#include <stdlib.h>

struct list {

  short  value;

  int id;

  struct list *next;

};

int menuPrint();

struct list * createList();

struct list *insertNodeList(int i,short val, struct list **head);

struct list *deleteNodeList(struct list **head);
```

```
void printNode(struct list *head);
int printNumOfNodes(struct list *head);
void printAddressNode(struct list *head);
void printAddressList(struct list * head);
void printSpecialNode(struct list* head);
void printSizeList(int help,struct list* head);
void printSizeNode();
void printFunctionAddress();
void printAll(struct list *head,int i);
int main(void){

   struct list *node = NULL;
   int choice;
   int num;
   int i,j=0,b;
   short v;

   int(*foo)(void);
   foo=&menuPrint;

   struct list* (*foo1)(void);
   foo1=&createList;

   struct list* (*foo2)(int,short,struct list**);
   foo2=&insertNodeList;

   struct list* (*foo3)(struct list**);
```

```
foo3=&deleteNodeList;

void(*foo4)(struct list*);
foo4=&printNode;

int(*foo5)(struct list*);
foo5=&printNumOfNodes;

void(*foo6)(struct list*);
foo6=&printAddressNode;

void(*foo7)(struct list*);
foo7=&printAddressList;

void(*foo8)(struct list*);
foo8=&printSpecialNode;

void(*foo9)(int,struct list*);
foo9=&printSizeList;

void(*foo10)();
foo10=&printSizeNode;

void(*foo11)();
foo11=&printFunctionAddress;

void(*foo12)(struct list*,int);
```

```
foo12=&printAll;


do{
        choice = foo();
        if(choice == 1){
                node = foo1();
        }
        else if(choice == 2){
                printf("Give value\n->");
        scanf("%d",&v);
        printf("Give ID\n->");
        scanf("%d",&i);
        foo2(i,v,&node);
                foo12(node,1);
        }
                else if(choice == 3){
                        foo3(&node);
                        foo12(node,1);
                }
                else if(choice == 4){
                        foo4(node);
                }
                else if(choice == 5){
                        b = foo5(node);
                        printf("The number of nodes is %d\n",b);
                }
                else if(choice == 6){
```

```
                foo6(node);
            }
            else if(choice == 7){
                foo7(node);
            }
            else if(choice == 8){
                foo8(node);
            }
            else if(choice == 9){
                b = foo5(node);
                foo9(b,node);
            }
            else if(choice == 10){
                foo10();
            }
            else if(choice == 11){
                foo11();
            }
            }while(choice != 12);

    system("pause");
        return 0;
}

int menuPrint(){
        int ch;
        printf("\n\n\nType (1) to create list\n");
```

```
        printf("Type (2) to insert a node\n");
        printf("Type (3) to delete the 1st node\n");
        printf("Type (4) to print special item\n");
        printf("Type (5) to print the number of items\n");
        printf("Type (6) to print special address of item\n");
        printf("Type (7) to print the address of list\n");
        printf("Type (8) to print the address of special item section\n");
        printf("Type (9) to print the size of list\n");
        printf("Type (10) to print the size of item\n");
        printf("Type (11) to print the addresses of functions\n");
        printf("Type (12) to exit\nYour choice: ");

        scanf("%d",&ch);

        return ch;
}

struct list *createList(){
                                    //1
         struct list *node = NULL;
         return node;
}

struct list *insertNodeList(int i,short val, struct list **head){                    //2

  struct list *node;
  struct list **node2 = head;
  node = (struct list *)malloc(sizeof(struct list));
```

```
  node->value = val;
  node->id = i;
  node->next = NULL;

  while (*node2 != NULL){

                node2 = &((**node2).next);
  }

  *node2 = node;
        return *head;
}

struct list * deleteNodeList(struct list **head) {                    //3

 if (*head != NULL){
   *head = (*head)->next;
 }
 else{
        printf("Empty List!\n");
        return;
 }
 return *head;
}

void printNode(struct list *head){
                //4
```

```
	struct list* node = head;
	int s;
	printf("Give node\n->");
	scanf("%d",&s);
	int i=0,flag = 0;
	while(i<s && node->next != NULL){
		if((i == (s-1)) && (node != NULL)){
			printf("Node info:\n\nValue: %d\nID: %d\n",node->value,node->id);
			flag = 1;
		}
		i++;
		node = node->next;
	}
	if(flag == 0){
		printf("This node does not exist\n");
	}
	return;
}

int printNumOfNodes(struct list *head){
	//5
	struct list* node = head;
	int i=0;
	while(node != NULL){
		i++;
		node = node->next;
	}
	return i;
```

```
}

void printAddressNode(struct list *head){
	//6

	struct list* node = head;
	int s;
	printf("Give node\n");
	scanf("%d",&s);
	int i=0;
	while(i<s){
		if((i == (s-1)) && (node != NULL)){
			printf("Node info:\nValue: %d\nID: %d\n",node->value,node->id);
			printf("Addresses:\nValue: %x\nID: %x\n",&node->value,&node->id);
			i++;
		}
		node = node->next;
	}
	return;
}

void printAddressList(struct list * head){		//7
	struct list* node = head;
	printf("Address of list: %x\n",head);
}

void printSpecialNode(struct list* head){
	//8
```

```
struct list* node = head;
int s1;
char ch;
short s2;


printf("Value or ID?(v-i)\n");
scanf("\n%c",&ch);
if(ch == 'i'){
        printf("Give id\n");
        scanf("%d",&s1);
        while(node != NULL){
                if(node->id == s1){
                        printf("ID address: %x\n",&node->id);
                }
                node = node->next;
        }
}
else if (ch == 'v'){
        printf("Give value\n");
        scanf("%d",&s2);
        while(node != NULL){
                if(node->value == s2){
                        printf("Value address: %x\n",&node->value);
                }
                node = node->next;
        }
}
```

```
        return;
}

void printSizeList(int help,struct list* head){        //9
struct list* node=head;
struct list *temp=head;
struct list *temp2;
int first,last,sizel;
        if(help == 0 || help == 1){
                printf("Size of list: %d bytes\n",2*sizeof(struct list));
        }
        else{
                first = (int)(node);
                last = (int)(node->next);
                printf("Size of list: %d\n",help*(last-first));

        }
}

void printSizeNode(){
                                        //10
        printf("Size of node: %d\n",sizeof(struct list));
}

void printFunctionAddress(){
        printf("Function addresses:\n\n_____\n\n");
        printf("CreateList: %x\n",createList);
        printf("InsertNodeList: %x\n",insertNodeList);
```

```
	printf("DeleteNodeList: %x\n",deleteNodeList);
	printf("PrintNode: %x\n",printNode);
	printf("PrintNumOfNodes: %x\n",printNumOfNodes);
	printf("PrintAddressNode: %x\n",printAddressNode);
	printf("PrintAddressList: %x\n",printAddressList);
	printf("PrintSpecialNode: %x\n",printSpecialNode);
	printf("PrintSizeList: %x\n",printSizeList);
	printf("PrintSizeNode: %x\n",printSizeNode);
	printf("PrintFunctionAddress: %x\n",printFunctionAddress);
	printf("PrintAll: %x\n_____________",printAll);
	return;
}
void printAll(struct list *head,int i){
	if(head == NULL){
		return;
	}
	else {
		printf("Node %d:\n",i);
		printf("Value: %d\nID: %d\n\n",head->value,head->id);
		printAll(head->next,i+1);
	}
	return;
}
```

Κώδικας σε γλώσσα Clang

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <setjmp.h>

#ifndef JAL
   #define jal(X) if(!setjmp(buf)) goto *X;
#endif
#ifndef JR31
   #define JR31 (longjmp(buf,1))
#endif

struct list {
  short  value;
  int id;
  struct list *next;
};

int menuPrint();
struct list *createList();
struct list *insertNodeList();
struct list *deleteNodeList();
void printNode();
void printAll(struct list *head,int i);

int R0=0,R1,R2,R3,R4=0,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16;
int R17,R18,R19,R20,R21=1,R22,R23,R24,R25,R26,R27,R28,R29,R30,R31;

static jmp_buf buf;
int main(void){
```

```
        short v;


        int(*foo)(void);
foo=&menuPrint;
struct list* (*foo1)(void);
foo1=&createList;
struct list* (*foo2)(void);
foo2=&insertNodeList;
struct list* (*foo3)(void);
foo3=&deleteNodeList;
void(*foo4)(void);
foo4=&printNode;

do_label:

                R15=(int)foo;
        jal(R15);
        if(R2 != 1) goto else_label_1;
                R16=(int)foo1;
                jal(R16);

                goto after_cond;
                else_label_1:
                if(R2 != 2) goto else_label_2;
                        printf("Give value\n->");
        scanf("%hd",&v);
```

```
            R6 = (int)v;
            printf("Give ID\n->");
            scanf("%d",&R5);
            R17=(int)foo2;
            jal(R17);
            printAll((struct list*)R4,1);
            goto after_cond;
            else_label_2:
            if(R2 != 3) goto else_label_3;
                        R18 = (int)foo3;
                        jal(R18);
                        printAll((struct list*)R4,1);
            goto after_cond;
            else_label_3:
            if(R2 != 4) goto after_cond;
                        R19=(int)foo4;
                        jal(R19);
            after_cond:
                        if(R2 == 5) goto after_loop;
    goto do_label;
    after_loop:



    return (EXIT_SUCCESS);
}


int menuPrint(){
```

```
            int R8;
            printf("\n\nType (1) to create list\n");
            printf("Type (2) to insert a node\n");
            printf("Type (3) to delete the 1st node\n");
            printf("Type (4) to print special item\n");
            printf("Type (5) to exit (5)\nYour choice: ");
            scanf("%d",&R8);
            R2 = R8;
            JR31;
}

struct list * createList(){
             struct list *node;

             node = NULL;
             R4 = (int)(NULL);
             R2 = R4;
             JR31;

}

 struct list *insertNodeList(){
   struct list *node, *returnValue;
   struct list *node2;

   node2=(struct list*)R4;
```

```
  node = (struct list *)malloc(sizeof(struct list));

  R27 = (int)NULL;
  node->value = R6;
  node->id = R5;
  node->next = NULL;

  if(R4 != (int)NULL) goto after_cond;
    returnValue = node;
    R4 = (int)node;
            R2 = R4;
    JR31;
  after_cond:

  while_label:
           if(node2->next == NULL) goto after_loop;
       node2 = node2->next;
       goto while_label;
    after_loop:

  node2->next = node;
  R2 = R4;
  JR31;
 }
struct list * deleteNodeList() {
  struct list *node;
  node = (struct list*)R4;
```

```
 if (node == NULL) goto else_label;
   node= (node)->next;
 goto after_cond;
 else_label:
          printf("Empty List!\n");
          JR31;
 after_cond:
 R4 = (int)node;
 R2 = R4;
 JR31;
}
void printNode(){

          struct list *node;
          node = (struct list*)R4;
          int R10,R30=0;
          printf("Give node\n->");
          scanf("%d",&R10);
          R31 = 0;
          while_label:
          if((R30 >= R10) || node->next == NULL) goto after_loop;
                    if(!(R30 == (R10-1)) && (node != NULL)) goto after_cond;
                              printf("\nNode info:\nValue: %d\nID: %d\n\n",node->value,node-
>id);
                              R31 = 1;
                    after_cond:
                    R30++;
                    node = node->next;
```

```
		goto while_label;
		after_loop:
		if(R31 == 1) goto after_if;
			printf("This node does not exist!\n");
		after_if:
		JR31;
}
void printAll(struct list *head,int i){
		if(head == NULL){
			return;
		}
		else {
			printf("Node %d:\n",i);
			printf("Value: %d\nID: %d\n\n",head->value,head->id);
			printAll(head->next,i+1);
		}
		return;
}
```

Κώδικας σε γλώσσα Assembly:

```
.data
.globl menu
.globl repint
menu: .asciiz "Type (1) to create list\n
Type (2) to insert a node\n
Type (3) to delete the 1st node\n
Type (4) to print special item\n
Type (5) to exit\n"
```

```
bye: .asciiz "Program terminated. Bye!"
askint: .asciiz "Please type your choice: "
repint: .asciiz "Your choice is : "
newline: .asciiz "\n"

.text
.globl main
main:

li $t1, 5                    # initialization exit register

li $v0, 4                    # Print Menu
la $a0, menu
syscall

loop:

li  $v0, 4                   # print new line
la  $a0, newline
syscall

li $v0, 4                    # print string
la $a0, askint
syscall

li  $v0, 5                   # read integer
```

```
syscall
add  $t0, $v0,$zero



li $v0, 4                  # print string
la $a0, repint
syscall



li  $v0, 1                 # print integer
move $a0, $t0
syscall

li  $v0, 4                 # print new line
la  $a0, newline
syscall

bne $t0, $t1,loop              # loop condition and in case of true goto loop label



li  $v0, 4                 # print new line
la  $a0, newline
syscall
```

```
li $v0, 4                    # print string
la $a0, bye
syscall

li  $v0, 10                  #exit of program
syscall
```