13/11/2015

Αναφορά Εργαστηρίου 3

Ομάδα

Κολομβάκης Χρήστος (Α.Μ.: 2013030103)
Ζαχαριουδάκης Χρήστος (Α.Μ.: 2014030056)

Προεργασία

Κατά την προεργασία του εργαστηρίου μας ζητήθηκε η μετατροπή του προγράμματος , που υλοποιήσαμε στο προηγούμενο εργαστήριο σε γλώσσα προγραμματισμού C,ώστε να εκτελεί τις ακόλουθες λειτουργίες :

1. Η κλήση των συναρτήσεων γίνεται πλέον με functionpointers.
2. Ουπολογισμός του μεγέθους της λίστας γίνεται πλέον με άθροιση των διαφορών των διευθύνσεων διαδοχικών κόμβων της λίστας .
3. Η υλοποίηση μιας συνάρτησης , η οποία εκτυπώνει την διεύθυνση κάθε συνάρτησης του προγράμματος , εκτός από την main () .

Επιπλέον , μας ζητήθηκε η μετατροπή του προγράμματος , που υλοποιήσαμε στο προηγούμενο εργαστήριο σε γλώσσα προγραμματισμού CLANG,ώστε να εκτελεί τις ακόλουθες λειτουργίες :

1. Η κλήση των συναρτήσεων θα γίνεται με functionpointers (χρήση των εντολών goto *(καταχωρητής με την διεύθυνση της συνάρτησης αποθηκευμένη ως ακέραιο), setjmp, longjmp)
2. Οι συναρτήσεις δεν έχουν ορίσματα και δεν επιστρέφουν τιμές , αλλά η τιμή επιστροφής αποθηκεύεται στον καταχωρητή R2 , και τα ορίσματα στους καταχωρητές R4 – R7 .

Τέλος μας ζητήθηκε η υλοποίηση ενός προγράμματος στη Assembly , όπου εκτυπώνεται το ίδιομενυμε αυτό του προγράμματος της CLANG , όπου ο χρήστης καλείται να επιλέξει τιμή από 1 εώς 4. Στην συνέχεια, η επιλογή του εκτυπώνεται , εκτυπώνεται ξανά το menu και ο χρήστης καλείται ξανά να επιλέξει . Αυτό θα γίνεται ατέρμονα μέχρι ο χρήστης επιλέξει 5,όπου και το πρόγραμμα θα τερματιστεί .

Περιγραφή Ζητούμενων

Ο σκοπός του 3ου εργαστηρίου ήταν η περαιτέρω εξοικείωση με την γλώσσα προγραμματισμού C στην χρήση των function pointers και την περαιτέρω κατανόηση της λειτουργίας της μνήμης του υπολογιστή .Συγκεκριμένα , κατανοήσαμε ότι τόσο τα δεδομένα όσο και ο κώδικας (και κατ’ επέκταση οι συναρτήσεις) καταλαμβάνουν κάποιο χώρο στην μνήμη , τον οποίο χρειάζονται για να χρησιμοποιηθούν από τον προγραμματιστή.

Επιπλέον , είχε σκοπό την εμβάθυνση στην γλώσσα προγραμματισμού CLANG , η οποία αποτελεί μια γλώσσα με λειτουργικότητα ενδιάμεση της γλώσσας υψηλού επιπέδου C και της γλώσσας χαμηλού επιπέδου Assembly . Στο τωρινό εργαστήριο απαιτήθηκε η χρήση καταχωρητών σε μορφή global μεταβλητών για την κλήση συναρτήσεων και για το πέρασμα και την επιστροφή ορισμάτων , και συγκεκριμένων καταχωρητών για κάθε εργασία . Η προσέγγιση αυτή προσομοίωσε ακόμα καλύτερα την Assembly, καθώς οι κλήση των συναρτήσεων γινόταν με παρόμοιο τρόπο, όπως και η διαχείριση των παραμέτρων . Συγκεκριμένα χρησιμοποιήσαμε συγκεκριμένες μεταβλητές- καταχωρητές στη CLANG για την τιμή επιστροφής (R2)και τις παραμέτρους των συναρτήσεων (R4-R7) ,με τον ίδιο τρόπο που στην Assembly χρησιμοποιούμε συγκεκριμένους καταχωρητές για αυτές τις εργασίες ($v0 - $v1 για επιστροφή τιμών και $a0 - $a3 για τις παραμέτρους των συναρτήσεων) . Με αυτό τον τρόπο εξοικειωθήκαμε καλύτερα με τις συμβάσεις της Assembly MIPS,οι οποίες βοηθούν στην δημιουργία ενός πιο δομημένου προγράμματος .

Περιγραφή της Εκτέλεσης

Η άσκηση έγινε με χρήση του περιβάλλοντος προγραμματισμού Netbeans. Συγκεκριμένα δημιουργήσαμε 2 νέα project της C (ένα εκ των οποίων στη CLANG), από το 'File →New Project → C/C++ → C/C++ Application - >Finish'

Έπειτα από το 'Run → Build Project' κάναμε compile το πρόγραμμα μας και από το 'Run →Run Project' , το εκτελέσαμε .

Μας ζητήθηκε να δημιουργήσουμε μια λίστα 5 κόμβων (nodes) στο πρόγραμμα της Cκαι Clang, και μετά να εκτελέσουμε τις λειτουργίες του προγράμματος .Επίσης έγινε δοκιμή των ελέγχων σφάλματος του προγράμματος , όπως εμφάνιση μηνύματος λάθους σε περίπτωση διαγραφή κόμβου σε μία κενή λίστα ή αίτησης εκτύπωσης στοιχείου που δεν υπάρχει στην λίστα .

Επιπλέον εκτελέσαμε το πρόγραμμα της Assembly στο πρόγραμμα PCSPIM , όπου εξετάστηκε η ορθή λειτουργία του και η γνώση μας στην εντολές τις Assembly. Τέλος, εξεταστήκαμε και στις λειτουργίες του προγράμματος PCSPIM , συγκεκριμένα στην χρήση breakpoints για τον εντοπισμό σφαλμάτων.

Παράδειγμα Εκτέλεσης:

C:

The software offers you the following options:
Please enter your choice (1-12):

1) Create a list.
2) Input a value at the end of the list.
3) Delete the first value of the list.
4) Print a value of the list.
5) Receive the number of values the list contains.
6) Print the memory address of a node.
7) Print the memory address of the list.

8) Print the memory address of the value of the node.
9) Print the size of the list in bytes.
10) Print the size of an element in bytes.
11) Print the address of a function.
12) Exit the software.

1
Enter the value of the element.
3

MENU
1
You have already created a list!

MENU
2
Enter the value of the element.
2

MENU
2
Enter the value of the element.
5

MENU
2
Enter the value of the element.
7

MENU
2
Enter the value of the element.
8

MENU
3

MENU
4
Enter which element of the list you want printed. (1 for the first and so on.)
1
The searched value is not found.

MENU
4

Enter which element of the list you want printed. (1 for the first, and so on.)
2
Id: 2 value: 2

MENU
5
The list contains 4 values.

MENU
6
Enter which element of the list you want its address printed. (1 for the first, and so on.)
2
Address of element: 284d0

MENU
6
Enter which element of the list you want its address printed. (1 for the first, and so on.)
1
The searched value is not found.

MENU
7
The address of the list is 284d0

MENU
8
Enter which element of the list you want printed. (1 for the first, and so on.)
2
Enter which field's address you want printed. 1 for id, 2 for value.
1
Address of id is: 284d0

MENU
9
The size of the list in bytes is:10615

MENU
10
The size of an element is : 16 bytes.

MENU
11
You can print the addresses of following functions :
1.createList()
2.insertNodeList()

3.deleteFirstNode()
4.numOfElements()
5.printElement()
6.printElementAddress()
7.printFieldAddress()
8.printListAddress()
9.printSizeOfElement()
10.printSizeOfList()
11.printFunction()
12.Exit the function printing menu
Choose the respective number of the function you wish to print :
3
The address of the deleteFirstNode() function is :4011d1 .

SUBMENU
11
The address of the printFunction () function is: 4014be.

SUBMENU
2
The address of the insertNodeList () function is: 40113f.

SUBMENU
12

MENU
12

CLANG:

The software offers you the following options :
Please enter your choice (1-5) :

1) Create a list.
2) Input a value at the end of the list.
3) Delete the first value of the list.
4) Print a value of the list.
5) Exit the software.

1
Enter the value of the element.
2

MENU
2

Enter the value of the element.
4

MENU
2
Enter the value of the element.
5

MENU
2
Enter the value of the element.
8

MENU
4
Enter which element of the list you want printed. (1 for the first, and so on.)
1
Id: 1 value: 2

MENU
3

MENU
4
Enter which element of the list you want printed. (1 for the first, and so on.)
1
The searched value is not found.

MENU
5

Assembly:

```
Printing the menu :

1) Create a list
2) Input a value at the end of the list.
3) Delete the first value of the list.
4) Print a value of the list.
5) Exit the software.:
The inserted value is :
1
You chose : 1

1) Create a list
2) Input a value at the end of the list.
3) Delete the first value of the list.
4) Print a value of the list.
5) Exit the software.:
The inserted value is :
2
You chose : 2

1) Create a list
2) Input a value at the end of the list.
3) Delete the first value of the list.
4) Print a value of the list.
5) Exit the software.:
The inserted value is :
3
You chose : 3

1) Create a list
2) Input a value at the end of the list.
3) Delete the first value of the list.
4) Print a value of the list.
5) Exit the software.:
The inserted value is :
4
You chose : 4

1) Create a list
2) Input a value at the end of the list.
3) Delete the first value of the list.
4) Print a value of the list.
5) Exit the software.:
The inserted value is :
5
You chose : 5
You have exited the software.
```

Συμπεράσματα

Κατανοήσαμε καλύτερα την λειτουργία της μνήμης μέσω της γλώσσας προγραμματισμού C. Μέσω της CLANG , συνειδητοποιήσαμε τις λειτουργίες που εκτελεί το σύστημα στο παρασκήνιο κατά τη σύνθεση και μεταγλώττιση γλωσσών προγραμματισμού υψηλού επιπέδου και εξοικειωθήκαμε με τις συμβάσεις που απαιτεί η ορθήλειτουργία ενός προγράμματος στην γλώσσα προγραμματισμού Assembly .

Από το πρόγραμμα που **αναπτύξαμε στην C** , συμπεράναμε ότι ο υπολογισμός του μεγέθους της λίστας προσθέτοντας τις διαφορές των διευθύνσεωντων διαδοχικών κόμβων της λίστας δεν είναι σωστός, ούτε ίδιος σε κάθε υπολογιστή. Αυτό συμβαίνει επειδή δεν υπάρχει καμία διαβεβαίωση ότι οι κόμβοι αποθηκεύονται διαδοχικά στη μνήμη. Επιπλέον ο τρόπος οργάνωσης της μνήμης διαφοροποιείται ανάμεσα σε διαφορετικά λειτουργικά συστήματα και hardware (όπως ο επεξεργαστής). Τέλος λόγω memoryfragmentation , ουπολογιστής υποχρεούται να καταλάβει μνήμη μη διαδοχικά. Ωστόσο ασχέτως σειράς , η δυναμικά δεσμευμένη μνήμη αποθηκεύεται στο heaptμήμα μνήμης , κάτι που επικυρώνεται από την εύρεση των διευθύνσεων των κόμβων .

Στο **πρόγραμμα της CLANG**είχαμε αρχικά ορίσει τους καταχωρητές ως int , όπως τα παραδείγματα που μας παρουσιάστηκαν στο φροντιστήριο , αλλά ο complilerNetbeansπαρουσίαζε πολλαπλά Warningsκαι υπήρχαν προβλήματα και στην εκτέλεση του προγράμματος . Για να τα επιλύσουμε , ορίσαμε τους καταχωρητές ως **longint** . Αυτό ήταν απαραίτητο επειδή αναπτύχθηκε σε σύστημα με επεξεργαστή 64 bit , με αποτέλεσμα οι δείκτες (διευθύνσεις) να έχουμε μέγεθος 8 byteέναντι των 4 byte , που καταλαμβάνουν σε σύστημα με επεξεργαστή 32bit . Έτσι ώστε να μην υπάρχει απώλεια πληροφορίας με την αποθήκευση δεδομένων δεικτών με castingστους καταχωρητές , τους ορίσαμε ως long (sizeof(long) = 8 , sizeof(int)= 4) . Από αυτό συμπεράναμε ότι η αρχιτεκτονική του επεξεργαστή ενός υπολογιστή προκαλεί διαφοροποιήσεις στην διαχείριση της μνήμης , κάτι που έγινε φανερό και στο πρόγραμμα της Cκαι σε προηγούμενα εργαστήρια και συγκεκριμένα στο μέγεθος των διευθυνσεων (δεικτών).

Τέλος στο **πρόγραμμα της Assembly** ,καταλάβαμε καλύτερα πως λειτουργούν οι εντολές li (load immediate) , la (load address) σε συνδυασμό με syscall (κλήση συστήματος) , για την αλληλεπίδραση με τον χρήστη . Συγκεκριμένα , αποθηκεύουμε ακέραιους στον καταχωρητή $v0 , την διεύθυνση αυτού που θέλουμε να χρησιμοποιήσουμε (να τυπώσουμε , να αλλάξει ο χρήστης εισάγοντας από το πληκτρολόγιο κτλ) στον καταχωρητή $a0 . Έπειτα η syscallδιαβάζει τα περιεχόμενα του $v0 , και ανάλογα με τον ακέραιο που είναι αποθηκευμένος διαβάζει τον $a0 και εκτελεί την αντίστοιχη λειτουργία. Επίσης μάθαμε να χρησιμοποιούμε τα labels και τις εντολές τύπου branchγια να αλλάζουμε την ροή του προγράμματος , το οποίο χρειάζεται στις κλήσεις συναρτήσεων και σε loops .

Παράρτημα - Κώδικας

A) ***Υλοποίηση σε C***

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

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

struct list *createList (int v, short int value) // Create a list with a node
{
```

```
        struct list * head ;

        head = (struct list *) malloc (sizeof (struct list));
        head->id = v;
        head->value = value;
        head->next = NULL;
        if (head!= NULL) return head ;
        else printf ("Operation failed\n");
}

 struct list *insertNodeList(int v,short int value, struct list *head) {  // Input a node at the end
of the list
  struct list *node, *returnValue;
  struct list *node2 = head;
  node = (struct list *)malloc(sizeof(struct list));
  node->id = v;
  node->value = value;
  node->next = NULL;
  if (head == NULL) {
   returnValue = node;
   return returnValue;
  }
  while (node2->next != NULL)
      node2 = node2->next;
  node2->next = node;
  return head;
 }


struct list * deleteFirstNode(struct list *head) // ???????? ?????? ??? ???? ??? ?????? 3
{

struct list *ptr=head;
struct list *tmp;

if (ptr != NULL)
{
        tmp = ptr;
   ptr = ptr->next;
   free(tmp);
   return ptr ;
}
else
{
   printf ("The list is empty.\n");
   return NULL ;
}
}


int numOfElements (struct list *l) // Return the number of elements (nodes) that the list has
```

```
{

struct list * tmp = l ;
int counter = 0 ;

while (tmp!=NULL)
{
        counter++ ;
        tmp = tmp->next ;
}
return counter ;
}

void printElement(int id, struct list *ptr) // Searches a node with the specified id and prints its
id and value
{

 struct list * tmp = ptr ;

 while (tmp!=NULL )
 {
        if (tmp->id==id){
    printf("Id: %d value: %d\n", tmp->id, tmp->value);
    return;
  }
        else tmp = tmp->next ;
 }

 printf ("The searched value is not found.\n");

}

void printElementAddress(int id, struct list *ptr){ // Searches a node with the specified id and
prints its address

 struct list * tmp = ptr ;

 while (tmp!=NULL )
 {
        if (tmp->id==id){
    printf("Address of element: %x\n", tmp);
    return;
  }
        else tmp = tmp->next ;
 }

 printf ("The searched value is not found.\n") ;
}

void printFieldAddress(int id, struct list *ptr) // Searches a node with the specified id and
prints the address of one of its fields
```

```
{

struct list * tmp = ptr ;
int choice;

 while (tmp!=NULL )
 {
          if (tmp->id==id){
    printf("Enter which field's address you want printed. 1 for id, 2 for value.\n");
    scanf("%d", &choice);
    if(choice==1)
          {
       printf("Address of id is: %x", &(tmp->id));
    }
    else if(choice==2){
       printf("Address of value is: %x", &(tmp->value));
    }
    else{
       printf("Invalid choice.\n");
    }
    return;

  }
          else tmp = tmp->next ;
 }

 printf ("The searched value is not found.\n");

}

void printListAddress (struct list *head) //Prints the address of the list
{
          printf ("The address of the list is : %x.\n",head);
}

void printSizeOfElement () // Prints the size of a node 10
{
          printf ("The size of an element is : %d bytes.\n",sizeof(struct list));
     return         ;
}

void printSizeOfList (struct list * head) // Prints the size of list in bytes
{
   printf ("\nThe size of the list is : %d bytes.\n",sizeof(struct list)*numOfElements(head));

   struct list * tmp = head ;
   long long sum=0;
   for(; tmp->next!=NULL; tmp=tmp->next){
      sum = sum + (long long) (tmp->next - tmp);
}
   printf("\nThe size of the list in bytes is:%d", sum);
```

```
}

struct list * (*createListPtr) (int, short int);
void (*printSizeOfElementPtr) () ;
void (*printSizeOfListPtr) (struct list * );
void (*printListAddressPtr) (struct list *);
void (*printFieldAddressPtr)(int , struct list *);
void (*printElementAddressPtr)(int , struct list *);
void (*printElementPtr)(int , struct list *);
int (*numOfElementsPtr) (struct list *) ;
struct list * (*deleteFirstNodePtr)(struct list *);
struct list * (*insertNodeListPtr)(int,short int , struct list *);
 void (*printFunctionPtr)();

void printFunction()
{
   int choice ;

   printf ("You can print the addresses of following functions : \n");

   while(1)
   {
     printf ("1.createList() \n");
     printf ("2.insertNodeList() \n");
     printf ("3.deleteFirstNode() \n");
     printf ("4.numOfElements() \n");
     printf ("5.printElement()\n");
     printf ("6.printElementAddress()\n");
     printf ("7.printFieldAddress() \n");
     printf ("8.printListAddress()\n");
     printf ("9.printSizeOfElement() \n");
     printf ("10.printSizeOfList()\n");
     printf ("11.printFunction()\n");
     printf ("12.Exit the function printing menu");

     printf ("Choose the respective number of the function you wish to print :  \n");
     scanf ("%d", &choice);
     switch(choice)
     {
        case 1:
           printf ("The address of the createList() function is :%x .  \n",createListPtr);
           break ;
        case 2:
           printf ("The address of the insertNodeList() function is :%x .  \n",insertNodeListPtr);
           break ;
        case 3:
           printf ("The address of the deleteFirstNode() function is :%x .
\n",deleteFirstNodePtr);
           break ;
        case 4:
```

```
            printf ("The address of the numOfElements() function is :%x .
\n",numOfElementsPtr);
            break ;
        case 5:
            printf ("The address of the printElement() function is :%x .  \n",printElementPtr);
            break ;
        case 6:
            printf ("The address of the printElementAddress() function is :%x .
\n",printElementAddressPtr);
            break ;
        case 7:
             printf ("The address of the printFieldAddress() function is :%x .
\n",printFieldAddressPtr);
            break ;
        case 8:
            printf ("The address of the printListAddress() function is :%x .
\n",printListAddressPtr);
            break ;
        case 9:
             printf ("The address of the printSizeOfElement() function is :%x .
\n",printSizeOfElementPtr);
            break ;
        case 10:
             printf ("The address of the printSizeOfList() function is :%x .  \n",printSizeOfListPtr);
            break ;
        case 11:
            printf ("The address of the printFunction() function is :%x .  \n",printFunctionPtr);
            break ;
        case 12 :
            return ;
            break ;
        default :
            printf ("Please choose one of the following options.\n");
            break;
     }
   }
}

int main (){

struct list * head = NULL ;
int choice ,  id = 1 , list_created = 0 ;
short int value ;


printf ("The software offers you the following options : \n");
printf ("Please enter your choice (1-12) : \n\n");

while (1) // It will exit only if the user chooses 11
{
            printf ("\n1) Create a list.\n")      ; // User Menu
```

```
    printf ("2) Input a value at the end of the list.\n")          ;
    printf ("3) Delete the first value of the list.\n")    ;
    printf ("4) Print a value of the list.\n")    ;
    printf ("5) Receive the number of values the list contains.\n")      ;
    printf ("6) Print the memory address of a node.\n")        ;
    printf ("7) Print the memory address of the list.\n")        ;
    printf ("8) Print the memory address of the value of the node.\n")          ;
    printf ("9) Print the size of the list in bytes.\n")    ;
    printf ("10) Print the size of an element in bytes.\n")       ;
printf ("11) Print the address of a function.\n")        ;
    printf ("12) Exit the software.\n\n")       ;

    scanf ("%d", &choice);

    switch (choice)
    {
            case 1 :
                    if (!list_created) // If the list is not created
                    {
            createListPtr = &createList;
            printf("Enter the value of the element.\n");
            scanf("%hd", &value);
            list_created = 1 ; // Shows that a list is created .
                       head = createListPtr(id,value);
            id++;
                    }
                    else printf ("You have already created a list!\n");
                    break;
            case 2 :
                    if (list_created) // If the list is created
                    {
            insertNodeListPtr = &insertNodeList;
            printf("Enter the value of the element.\n");
            scanf("%hd", &value);
            head = insertNodeListPtr(id,value,head);
                       id++;
          }
                    else printf ("You must create a list first.\n")        ;
                    break;
            case 3 :
                    if (list_created)
        {
             deleteFirstNodePtr = &deleteFirstNode ;
                            head = deleteFirstNodePtr (head);
        }
                    else printf ("You must create a list first.\n")        ;
                    break;
            case 4 :
                    if (list_created)
                    {
          int temp;
```

```
                printElementPtr = &printElement ;
                        printf("Enter which element of the list you want printed. (1 for the
first, and so on.)\n");
                        scanf("%d", &temp);
                        printElementPtr(temp,head);
                      }
                      else printf ("You must create a list first.\n")       ;
                      break;
                case 5 :
                      if (list_created)
              {
                numOfElementsPtr = &numOfElements ;
                printf ("The list contains %d values.\n",numOfElementsPtr(head));
              }
                      else printf ("You must create a list first.\n")       ;
                      break;
                case 6 :
                      if (list_created)
                      {
                int temp;
                printElementAddressPtr = &printElementAddress ;
                        printf("Enter which element of the list you want its address printed.
(1 for the first, and so on.)\n");
                        scanf("%d", &temp);
                        printElementAddressPtr(temp,head);
                      }
                      else printf ("You must create a list first.\n")       ;
                      break;
                case 7 :
                      if (list_created)
                            printf ("The address of the list is %x", head );
                      else printf ("You must create a list first.\n");
                      break;
                case 8 :
                      if (list_created)
                      {
                int temp;
                printFieldAddressPtr = &printFieldAddress ;
                         printf("Enter which element of the list you want printed. (1 for the
first, and so on.)\n");
                         scanf("%d", &temp);
                printFieldAddressPtr(temp,head);
                      }
                      else printf ("You must create a list first.\n")       ;
                      break;
                case 9 :
                      if (list_created)
              {
                printSizeOfListPtr = &printSizeOfList;
                printSizeOfListPtr(head);
              }
```

```
                else printf ("You must create a list first.\n") ;
                        break;
                    case 10 :
                        if (list_created)
                {
                   printSizeOfElementPtr = printSizeOfElement;
                   printSizeOfElementPtr();
                }
                        else printf ("You must create a list first.\n")        ;
                break ;
            case 11:
                if (list_created)
                {
                   printFunctionPtr = printFunction ;
                   printFunctionPtr();
                 }
                else printf ("You must create a list first.\n") ;
                break;
                    case 12 :
                        return (EXIT_SUCCESS);
                    default :
                        printf ("Please choose one of the following options.\n");
                        break ;
        }
}
}
```

B) Υλοποίηση σε CLANG

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

long R0 = 0;

long R1 = 1, R2, R3 =2, R4 = 1, R5, R6, R7, R8 , R9 , R10, R11, R12, R13, R14 = 3, R15 = 4, R16 =
5, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27 = (long)NULL, R28, R29, R30, R31;
static jmp_buf buf ;



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

void createList () // Create a list with a node
{
```

```
    // struct list * head ;
        R24 = (long *) malloc (sizeof (struct list));
        struct list *head = (struct list*)R24;

        R27 = NULL;

        head->id = R4;
        head->value = R5;
        head->next = (struct list*) R27;
        R9 = (long)head;

        if (R9 == R27) goto elsecreate ;

R2  =  R9;
longjmp(buf,1);

        elsecreate :
        printf ("Operation failed\n");
longjmp(buf,1);
}

void insertNodeList() {  // Input a node at the end of the list

  R24 = (int*)malloc(sizeof(struct list));
struct list *node = (struct list*)R24;
struct list *node2 = (struct list *) R2;
node->id = R4;
node->value = R5;
node->next = (struct list*)R27;


if (node2 != ((struct list*)R27)) goto afterif ;
   R2 = R24;
longjmp(buf,1);

afterif :

whileinsert :

if  (node2->next == NULL)  goto afterloop ;

      node2 = node2->next;
goto whileinsert;

afterloop :
  node2->next = (struct list*)R24;
longjmp(buf,1);
 }


void deleteFirstNode()  // Delete the first node of the list
```

```
{

struct list *ptr = (struct list *) R2;
struct list *tmp;

if (ptr == ((struct list*)R27)) goto elsedelete ;

        tmp = ptr;
ptr = ptr->next;
free(tmp);
   R2 = (long) ptr ;
longjmp(buf,1);


elsedelete :
printf ("The list is empty.\n");
longjmp(buf,1);
}

void printElement ()  // Searches a node with the specified id and prints its id and value
{

struct list *tmp = (struct list * )R2;

whileprint :

if (tmp == ((struct list*)R27) ) goto afterprint_loop ;

        if (tmp->id != R7) goto elseprint_label ;

printf("Id: %d value: %d\n", tmp->id, tmp->value);
longjmp(buf,1);

        elseprint_label:
        tmp = tmp->next ;

        goto whileprint ;
afterprint_loop :

printf ("The searched value is not found.\n");
longjmp(buf,1);

}

int main () {

 //  R18 : choice ,  R8 : list_created , R4 : id , R5 : value  , R1 , R3 , R14 ,R15,  R16 are constants
defined above ,
 //  R2 returns head  , R20 call function, R9 has the value of head, R27 contains NULL

  // Function Pointers
```

```
void (*CreateList)(void) = &createList;
void (*InsertNodeList)(void) = &insertNodeList;
void (*PrintElement)(void) = &printElement;
void (*DeleteFirstNode)(void) = &deleteFirstNode;


printf ("The software offers you the following options : \n");
printf ("Please enter your choice (1-5) : \n\n");

while_label :

        printf ("\n1) Create a list.\n")     ; // User Menu
        printf ("2) Input a value at the end of the list.\n")          ;
        printf ("3) Delete the first value of the list.\n")    ;
        printf ("4) Print a value of the list.\n")    ;
        printf ("5) Exit the software.\n\n")         ;

        scanf ("%d", &R18);

        if (R18 < R1 ) goto else_label ;
        if (R18 > R16) goto else_label ;

                        if (R18 != R1) goto case2_label ; // case 1
                        if (R8 != R0) goto else1_label;
                        printf("Enter the value of the element.\n");
                        scanf("%d", &R5);

                         R8 = 1 ;

                         R20 = (long)CreateList;
                        if(!setjmp(buf)) goto  *R20 ;

                         R4++;
                        goto while_label ;
                        else1_label :
                        printf ("You have already created a list!\n");
                        goto while_label ;

                case2_label :

                        if (R18 != R3) goto case3_label ; // case 2
                        if (R8 == R0) goto else2_label ;

        printf("Enter the value of the element.\n");
        scanf("%d", &R5);


            R20 = (long)InsertNodeList;
        if(!setjmp(buf)) goto  *R20 ;
```

```
                R4++;
                goto while_label ;
                else2_label :
                printf ("You have not created a list!\n");
                goto while_label;

        case3_label :

                if (R18 != R14) goto case4_label ; // case 3
                if (R8 == R0) goto else3_label ;
                 R20 = (long)DeleteFirstNode;
                if(!setjmp(buf)) goto  *R20 ;

                goto while_label ;
                else3_label :
                printf ("You have not created a list!\n");
                goto while_label ;


        case4_label :

                if (R18 != R15) goto case5_label ; // case 4
                if (R8 == R0) goto else4_label ;

                printf("Enter which element of the list you want printed. (1 for the
                first, and so on.)\n");
                scanf("%d", &R7);
                R20 = (long)PrintElement;
                if(!setjmp(buf)) goto  *R20 ;


                goto while_label ;

                else4_label : // case 5
                printf ("You have not created a list!\n");
                goto while_label ;


        case5_label : return (0);

                else_label :        // default
                printf ("Please choose one of the following options.\n");

                goto while_label ;
        }
```

C) Υλοποίηση σε Assembly

```
.data
.globl askint
```

```
.globl repint
.globl Menu_Print
.globl Choice1
.globl Choice2
.globl Choice3
.globl Choice4
.globl Choice5
.globl newline
.globl ChoicePrint
.globl endword

Menu_Print: .asciiz "Printing the menu : "
Choice1: .asciiz "1) Create a list"
Choice2: .asciiz "2) Input a value at the end of the list."
Choice3: .asciiz "3) Delete the first value of the list."
Choice4: .asciiz "4) Print a value of the list."
Choice5: .asciiz "5) Exit the software.: "
repint: .asciiz "The inserted value is : "
ChoicePrint: .asciiz "You chose : "
newline: .asciiz "\n"
endword : .asciiz "You have exited the software. "

.text
.globl main
main:

li $v0, 4                              # 4 is the code to print a string
la $a0, Menu_Print  # a0 is the pointer to the string to # print and gets the value from #
"variable" Menu_Print
syscall

li $v0, 4
la $a0, newline

loopstart :

li $v0, 4
la $a0, newline
syscall

li $v0, 4
la $a0, newline
syscall


li $v0, 4                               # 4 is the code to print a string
la $a0, Choice1                      # "1) Create a list"
syscall

li $v0, 4
la $a0, newline
```

```
syscall

li $v0, 4                       # "2) Input a value at the end of the list."
la $a0, Choice2
syscall

li $v0, 4
la $a0, newline
syscall

li $v0, 4                       # "3) Delete the first value of the list."
la $a0, Choice3
syscall

li $v0, 4
la $a0, newline
syscall

li $v0, 4                        # "4) Print a value of the list."
la $a0, Choice4
syscall

li $v0, 4
la $a0, newline
syscall

li $v0, 4                       # "5) Exit the software.: "
la $a0, Choice5
syscall

li $v0, 4
la $a0, newline
syscall

li $v0, 4                       # "The inserted value is : "
la $a0, repint
syscall

li $v0, 4
la $a0, newline
syscall

li  $v0, 5  # 5 is the code to read an integer
syscall  # result is placed in $v0

move $s0 , $v0

li $v0, 4                       # "You chose : "
la $a0, ChoicePrint
syscall
```

```
li $v0, 1                    # 1 is the code to print an integer
move $a0,$s0
syscall

beq  $s0,5,exit
b loopstart

exit:

li $v0, 4
la $a0, newline
syscall


li $v0, 4                    # "You have exited the software. "
la $a0, endword
syscall

li  $v0, 10 # Exits the software
syscall
```

D) Χάρτης Μνήμης

Stack Memory Auto Variables Local Variables Function Parameters	**Local Variables** &id = 28ff3c &list_created = 28ff38 &choice = 28ff40
Static Memory Global Variables Static Variables Functions	**Functions** &createList () = 401290 &insertNode()= 4012e8 &printFunction() = 4015ca &main() = 4017d4
Heap Memory Dynamically Allocated Memory	**Nodes / List** Node 1/List Address = 871770 Node 2 = 873428 Node 5 = 873470