24/11/2015

# Αναφορά Εργαστηρίου 4

Ομάδα

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

## Προεργασία

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

1. Δημιουργία Λίστας με συγκεκριμένο αριθμό στοιχείων
2. Εισαγωγή στοιχείου στο τέλος της λίστας
3. Διαγραφή του τελευταίου στοιχείου της λίστας
4. Εκτυπώνει συγκεκριμένο στοιχείο της λίστας
5. Εκτυπώνει τον αριθμό στοιχείων της λίστας
6. Εκτυπώνει διεύθυνση συγκεκριμένου στοιχείου της λίστας
7. Εκτύπωση της διεύθυνσης της λίστας
8. Εκτύπωση του στοιχείου της λίστας με το μικρότερο (min)value . Όπου εκτυπώνεται η θέση , το idκαι το value .

Επιπλέον , μας ζητήθηκε η μετατροπή του νέου προγράμματος CLANG σε γλώσσα προγραμματισμού Αssembly με την διαφορά ότι στην Clang χρησιμοποιούμε λίστα , ενώ στην Assembly χρησιμοποιούμε στατικά δεδομένα (πίνακα δηλαδή, δεν χρησιμοποιήθηκε εντολή για δέσμευση μνήμης) , δηλαδή δεσμεύτηκαν 800 byte , αρκετός χώρος για 100 καταχωρήσεις . (Δεδομένου ότι το id και το value καταλαμβάνουν 8 byte στη μνήμη καθώς τους διαχειριστήκαμε ως ακεραίους (sizeof(int) = 4)

## Περιγραφή Ζητούμενων

Σκοπός του 4ου εργαστηρίου ήταν η περαιτέρω εξοικείωση με την γλώσσα προγραμματισμού CLANG στην εκτέλεση διάφορων λειτουργιών και της επεξεργασίας της μνήμης .Στην Clang ,όπως και στην Assembly τηρήσαμε αυστηρά τις συμβάσεις , δηλαδή την χρήση συγκεκριμένων καταχωρητών για συγκεκριμένες εργασίες .(Ορίσματα συναρτήσεων , τιμή επιστροφής συνάρτησης κ.τ.λ.).

Κυριότερος σκοπός του εργαστηρίου ήταν η επιτυχής μετατροπή και υλοποίηση του προγράμματος της Clang σε Assembly .Σε αυτό το εργαστήριο απαιτήθηκε η αυστηρότερη τήρηση των συμβάσεων της Assembly , καθώς και η χρήση πολύ περισσότερων εντολών και πιο πολύπλοκου κώδικα . Επίσης για πρώτη φορά μας ζητήθηκε η κλήση συναρτήσεων στην Assembly, όπου χρησιμοποιήσαμε τις συμβάσεις για το πέρασμα των παραμέτρων .Συγκεκριμένα περάσαμε τις τιμές που μας ενδιέφεραν από τους καταχωρητές $s\_ στους καταχωρητές $a\_στην main , έπειτα μέσα στην συνάρτηση περάσαμε τις τιμές από τους καταχωρητές $a\_σε καταχωρητές $t\_. Τέλος περάσαμε την τιμή επιστροφής στον καταχωρητή $v0 και έπειτα στην main περάσαμε την τιμή του $v0 σε ένα από τους καταχωρητές $s\_ .Βασικό μέρος του προγράμματος είναι η αλληλεπίδραση των καταχωρητών με την μνήμη , δηλαδή τον πίνακα που έχουμε δεσμεύσει σε αυτή. Αυτή γίνεται με τις εντολές loadκαι storeκαι την αυξομείωση του καταχωρητή που περιέχει την διεύθυνση του πρώτου στοιχείου του πίνακα. (Ένας προσωρινός καταχωρητής $t\_ στον οποίο έχει περαστεί η διεύθυνση, ενώ η διεύθυνση του πρώτου στοιχείου υπάρχει μόνιμα και σε ένα καταχωρητή $s\_). Όπως θα δείτε και παρακάτω, για την αποθήκευση της διεύθυνσης του πίνακα χρησιμοποιήθηκε ο καταχωρητής $s1 και ο $t1 για τροποποιήσεις εντός συναρτήσεων.Με χρήση αυτής της λογικής στην κλήση συναρτήσεων, μαζί με εντολές branch όπου υπήρχε ανάγκη , και εντολών load και store υλοποιήσαμε το πρόγραμμα μας .

## Περιγραφή της Εκτέλεσης

Η άσκηση έγινε με χρήση του περιβάλλοντος προγραμματισμούNetbeans. Συγκεκριμένα δημιουργήσαμε νέο project της Cαπό το ‘File →NewProject → C/C++ → C/C++ Application - >Finish’

Έπειτα από το ‘Run →Βuild Project’ κάναμε compileτο πρόγραμμα μας και από το ‘Run →RunProject’ , το εκτελέσαμε .

Μας ζητήθηκε να δημιουργήσουμε μια λίστα 5 κόμβων (nodes) στο πρόγραμμα της Clang, και μετά να εκτελέσουμε τις λειτουργίες του προγράμματος , με ιδιαίτερη έμφαση στις νέες λειτουργίες που προσθέσαμε σε σχέση με το προηγούμενο εργαστήριο , όπως η δημιουργία λίστας πολλαπλών κόμβων και η εύρεση της ελάχιστης τιμής . 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 Assemblyστο πρόγραμμα PCSPIM , όπου εξετάστηκε η ορθή λειτουργία του , η γνώση μας στην εντολές τις Assembly και η κατανόηση μας ως προς τις αλλαγές που προκαλεί η καθεμία εντολή στους καταχωρητές και στην μνήμη .

**Παράδειγμα Εκτέλεσης:**

**CLANG:**

The software offers you the following options :

Please enter your choice (1-9) :

1) Create a list.

2) Input a value at the end of the list.

3) Delete the last value of the list.

4) Print a value of the list.

5) Print the number of nodes.

6) Print the address of a node.

7) Print the address of the list.

8) Print the minimum value.

9) Exit the software.

1

Enter the number of nodes , you wish the list to have : 4

Enter the value of the first node : 1

Enter the value of the node : 2

Enter the value of the node : 3

Enter the value of the node : 4

**MENU**

2

Enter the value of the element.

5

**MENU**

4

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

1

Id: 1 value: 1

**MENU**

4

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

2

Id: 2 value: 2

**MENU**

4

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

5

Id: 5 value: 5

**MENU**

3

**MENU**

4

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

5

The searched value is not found.

**MENU**

5

The number of elements is : 4.

**MENU**

6

Enter which element of the list you want its address printed. (1 for the first, and so on.)

1

Address of element: 58460

**MENU**

6

Enter which element of the list you want its address printed. (1 for the first, and so on.)

5

The searched value is not found.

**MENU**

7

The address of the list is : 58460.

**MENU**

8

The node with the minimum value is :

Address : 361568

Id : 1

Value : 1

**MENU**

2

Enter the value of the element.

-1

**MENU**

8

The node with the minimum value is :

Address : 361696

Id : 6

Value : -1

**MENU**

9

You have exited the software!

**Assembly:**

Welcome. Enter a number from 1 to 9.

Press 1 to create a list with a specific number of values.

Press 2 to insert a value at the end of the list.

Press 3 to delete the last value of the list.

Press 4 to print a value of the list.

Press 5 to print the total number of nodes.

Press 6 to print the address of a node.

Press 7 to print the address of the list.

Press 8 to print the minimum value of the list.

Press 9 to exit the software.

1

Enter the number of nodes you want the list to have.

4

4 values will be inserted.

Enter the value to be inserted.

1

Enter the value to be inserted.

2

Enter the value to be inserted.

3

Enter the value to be inserted.

4

Welcome. Enter a number from 1 to 9.

**MENU**

2

Enter the value to be inserted.

5

Welcome. Enter a number from 1 to 9.

**MENU**

4

Enter which element you want printed. (1 for the first and so on...)

1

Id is : 1

Value is : 1

Welcome. Enter a number from 1 to 9.

**MENU**

4

Enter which element you want printed. (1 for the first and so on...)

5

Id is : 5

Value is : 5

Welcome. Enter a number from 1 to 9.

**MENU**

3

**MENU**

4

Enter which element you want printed. (1 for the first and so on...)

5

The requested node does not exist.

Welcome. Enter a number from 1 to 9.

**MENU**

5

The total number of values of the list is: 4

Welcome. Enter a number from 1 to 9.

**MENU**

6

Enter which value's address you want printed. (1 for the first and so on...)

1

The address of the variable is: 268502100

Welcome. Enter a number from 1 to 9.

**MENU**

6

Enter which value's address you want printed. (1 for the first and so on...)

2

The address of the variable is: 268502108

Welcome. Enter a number from 1 to 9.

**MENU**

7

The address of the list is: 268502100

Welcome. Enter a number from 1 to 9.

**MENU**

8

The address of the lowest value is:268502100

The id of the lowest value is:1

The lowest value is: 1

Welcome. Enter a number from 1 to 9.

**MENU**

2

Enter the value to be inserted.

-1

Welcome. Enter a number from 1 to 9.

**MENU**

8

The address of the lowest value is:268502132

The id of the lowest value is:6

The lowest value is: -1

Welcome. Enter a number from 1 to 9.

**MENU**

9

## Συμπεράσματα

Μέσω της CLANG , συνειδητοποιήσαμε τις λειτουργίες που εκτελεί το σύστημα στο παρασκήνιο κατά τη σύνθεση και μεταγλώττιση γλωσσών προγραμματισμού υψηλού επιπέδου και εξοικειωθήκαμε περαιτέρω με τις συμβάσεις που απαιτεί η ορθήλειτουργία ενός δομημένου προγράμματος στην γλώσσα προγραμματισμού Assembly .

Στο **πρόγραμμα της CLANG** προσθέσαμε επιπλέον λειτουργίες και αλλάξαμε κάποιες από τις υπάρχοντες σύμφωνα με τις ζητούμενες προδιαγραφές, χρησιμοποιώντας τις συμβάσεις και τις εντολές με τις οποίες εξοικειωθήκαμε σε προηγούμενα εργαστήρια και τις γνώσεις μας από την γλώσσα προγραμματισμού C για την διαχείριση της συνδεδεμένης λίστας δεδομένων .

Τέλος στo **πρόγραμμα της Assembly** ,αποκτήσαμε μεγαλύτερη εμπειρία στην υλοποίηση προγραμμάτων , χρησιμοποιώντας τις εντολές και τις συμβάσεις της γλώσσας για την υλοποίηση ενός δομημένου προγράμματος . Ιδιαίτερη σημαντική ήταν η εμπειρία που αποκτήσαμε στην κλήση συνάρτησης με των κατάλληλων εντολών και καταχωρητών . Θεωρήσαμε ενδιαφέρον το εύρος των δυνατοτήτων που προσφέρει η Assembly, όπως την δυνατότητα υλοποίησης ενός αρκετά μεγάλου και περίπλοκου προγράμματος εξίσου αποτελεσματικά με μια γλώσσα προγραμματισμού υψηλού επίπεδου (C/CLANG) . Τέλος αξιοσημείωτη είναι η αμεσότητα με την οποία ο προγραμματιστής μπορεί να διαχειριστεί την μνήμη μέσω των οδηγιών για δέσμευση μνήμης και των εντολών load και store.

## Παράρτημα - Κώδικας

1. ***Υλοποίηση σε CLANG***

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <setjmp.h>*

*long long int R0 = 0;*

*long long int R1 = 1, R2 = 0, R3 =2, R4 = 1, R5 = 0 , R6 = 0, R7 = 0, R8 = 0 , R9 , R10, R11, R12, R13, R14 = 3, R15 = 4, R16 = 9, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27 = NULL, R28, R29, R30, R31;*

*static jmp\_buf buf ;*

*struct list*

*{*

*int id ;*

*short int value ;*

*struct list \* next ;*

*};*

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

*R25 = (long long) malloc(sizeof(struct list));*

*((struct list \*)R25)->id = (int) ++R4;*

*((struct list \*)R25)->value = (short int) R5;*

*((struct list \*)R25)->next = (struct list\*) R27;*

*if (R6 != (long long)((struct list\*)R27)) goto afterif ;*

*R2 = R25; // Return value*

*longjmp(buf,1);*

*afterif :*

*whileinsert :*

*if (((struct list \*)R6)->next == NULL) goto afterloop ;*

*R6 = (long long)((struct list \*)R6)->next;*

*goto whileinsert;*

*afterloop :*

*((struct list \*)R6)->next = (struct list\*)R25;*

*longjmp(buf,1);*

*}*

*void (\*InsertNodeList)(void) = &insertNodeList;*

*void createList () // Create a list with a node*

*{*

*if (R22 == R0)*

*{*

*printf ("You have not created a list");*

*longjmp(buf,1);*

*}*

*else if (R22 < R0)*

*{*

*printf ("Error! Enter a positive integer!\n");*

*longjmp(buf,1);*

*}*

*else*

*{*

*R8 = 1 ; // A list is created*

*R6 = (long long) (malloc (sizeof(struct list)));*

*((struct list \*)R6)->id = (int) R4;*

*((struct list \*)R6)->value = (short int) R5;*

*((struct list \*)R6)->next = (struct list\*) R27;*

*--R22 ;*

*R2 = R6 ;*

*longjmp(buf,1);*

*}*

*if (R6 == (long long) NULL) goto elsecreate ;*

*R2 = R6 ;*

*longjmp(buf,1);*

*elsecreate :*

*printf ("Operation failed\n");*

*longjmp(buf,1);*

*}*

*void deleteLastNode() // Delete the last node of the list*

*{*

*if ((R6 != (long long) NULL) && (((struct list \*)R6)->next==NULL))*

*{*

*free((struct list \*)R6);*

*R2 = (long long) NULL; // return value*

*}*

*else if (R2 == (long long) NULL)*

*{*

*printf ("The list is empty.\n");*

*}*

*else*

*{*

*while(((struct list \*)R6)->next != NULL)*

*{*

*R28 = R6;*

*R6= (long long)((struct list \*)R6)->next;*

*}*

*free((struct list \*)R6);*

*((struct list \*)R28)->next=NULL;*

*}*

*longjmp(buf,1);*

*}*

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

*{*

*whileprint :*

*if (R6 == (long long) NULL ) goto afterprint\_loop ;*

*if (((struct list \*)R6)->id != R7) goto elseprint\_label ;*

*printf("Id: %d value: %d\n", ((struct list \*)R6)->id,((struct list \*)R6)->value);*

*longjmp(buf,1);*

*elseprint\_label:*

*R6 = (long long) ((struct list \*)R6)->next ;*

*goto whileprint ;*

*afterprint\_loop :*

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

*longjmp(buf,1);*

*}*

*void numOfElements ()*

*{*

*R22 = 0 ;*

*while (R6 != (long long) NULL)*

*{*

*R22++ ;*

*R6 = (long long) ((struct list \*)R6)->next ;*

*}*

*printf ("The number of elements is : %ld.\n",R22);*

*longjmp(buf,1);*

*}*

*void nodeAddress ()*

*{*

*while (R6 != (long long) NULL )*

*{*

*if (((struct list \*)R6)->id==R7)*

*{*

*printf("Address of element: %x\n", R6);*

*longjmp(buf,1);*

*}*

*else R6 = (long long) ((struct list \*)R6)->next ;*

*}*

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

*longjmp(buf,1);*

*}*

*void listAddress ()*

*{*

*printf ("The address of the list is : %x.\n",R2);*

*longjmp(buf,1);*

*}*

*void minimumValue ()*

*{*

*if (R2 == (long long) NULL)*

*{*

*printf ("The list is empty.\n");*

*longjmp(buf,1);*

*}*

*else*

*{*

*while (R6 != (long long) NULL)*

*{*

*if ((((struct list \*)R7)->value) > ((struct list \*)R6)->value) R7 = R6 ;*

*R6 = (long long) ((struct list \*)R6)->next ;*

*}*

*printf ("The node with the minimum value is : \n");*

*printf ("Address : %d\n",((struct list \*)R7));*

*printf ("Id : %d\n",((struct list \*)R7)->id);*

*printf ("Value : %d\n",((struct list \*)R7)->value);*

*longjmp(buf,1);*

*}*

*longjmp(buf,1);*

*}*

*int main () {*

*// R18 : choice , R8 : list\_created , R4 : id , R5 : value ,R6 head as parameter ,R7 id choice / tmpHead , R0 , R1 , R3 , R14 ,R15, R16 constants*

*// R2 returns head , R20 call function , R22 : numofElements , R24 MinValue Node , R25 newNode , R27 NULL*

*// Function Pointers*

*void (\*CreateList)(void) = &createList;*

*void (\*PrintElement)(void) = &printElement;*

*void (\*DeleteLastNode)(void) = &deleteLastNode;*

*void (\*NumOfElements)(void) = &numOfElements;*

*void (\*NodeAddress)(void) = &nodeAddress;*

*void (\*ListAddress)(void) = &listAddress;*

*void (\*MinimumValue)(void) = &minimumValue;*

*printf ("The software offers you the following options : \n");*

*printf ("Please enter your choice (1-9) : \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 last value of the list.\n") ;*

*printf ("4) Print a value of the list.\n") ;*

*printf ("5) Print the number of nodes.\n") ; // User Menu*

*printf ("6) Print the address of a node.\n") ;*

*printf ("7) Print the address of the list.\n") ;*

*printf ("8) Print the minimum value.\n") ;*

*printf ("9) Exit the software.\n\n") ;*

*scanf ("%Ld",&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 number of nodes , you wish the list to have : ");*

*scanf ("%Ld",&R22);*

*if (R22 <= R0) goto aftermainif ;*

*printf ("Enter the value of the first node : ");*

*scanf ("%Ld",&R5); // First Node*

*aftermainif :*

*R6 = R2; // Head Parameter*

*R20 = (long long)CreateList;*

*if(!setjmp(buf)) goto \*R20;*

*whilemain :*

*if ((R22--) == R0) goto after\_while ;*

*printf ("Enter the value of the node : ");*

*scanf ("%Ld",&R5); // First Node*

*R6 = R2; // Parameter*

*R20 = (long long)InsertNodeList;*

*if(!setjmp(buf)) goto \*R20 ;*

*goto whilemain ;*

*after\_while :*

*goto while\_label ;*

*else1\_label :*

*printf ("You have already created a list!\n");*

*goto while\_label ;*

*case2\_label : // case 2*

*if (R18 != R3) goto case3\_label ;*

*if (R8 == R0) goto else2\_label ;*

*printf("Enter the value of the element.\n\n");*

*scanf("%Ld", &R5); // Parameter*

*R6 = R2;*

*R20 = (long long)InsertNodeList;*

*if(!setjmp(buf)) goto \*R20 ;*

*goto while\_label ;*

*else2\_label :*

*printf ("You have not created a list!\n\n");*

*goto while\_label;*

*case3\_label : // case 3*

*if (R18 != R14) goto case4\_label ;*

*if (R8 == R0) goto else3\_label ;*

*R6 = R2; // Parameter*

*R20 = (long long)DeleteLastNode;*

*if(!setjmp(buf)) goto \*R20 ;*

*goto while\_label ;*

*else3\_label :*

*printf ("You have not created a list!\n\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("%Ld", &R7); // Parameter*

*R6 = R2;*

*R20 = (long long)PrintElement;*

*if(!setjmp(buf)) goto \*R20 ;*

*goto while\_label ;*

*else4\_label : // case 5*

*printf ("You have not created a list!\n\n");*

*goto while\_label ;*

*case5\_label :*

*if (R18 != 5) goto case6\_label ; // case 5*

*if (R8 == R0) goto else5\_label ;*

*R6 = R2 ; // Parameter*

*R20 = (long long)NumOfElements;*

*if(!setjmp(buf)) goto \*R20 ;*

*goto while\_label ;*

*else5\_label : // case 5*

*printf ("You have not created a list!\n\n");*

*goto while\_label ;*

*case6\_label : // case 6*

*if (R18 != 6) goto case7\_label ;*

*if (R8 == R0) goto else6\_label ;*

*printf("Enter which element of the list you want its address printed. (1 for the first, and so on.)\n");*

*scanf("%Ld", &R7); // Parameters*

*R6 = R2 ;*

*R20 = (long long)NodeAddress;*

*if(!setjmp(buf)) goto \*R20 ;*

*goto while\_label ;*

*else6\_label : // case 5*

*printf ("You have not created a list!\n\n");*

*goto while\_label ;*

*case7\_label : // case 7*

*if (R18 != 7) goto case8\_label ;*

*if (R8 == R0) goto else7\_label ;*

*R6 = R2;*

*R20 = (long long)ListAddress;*

*if(!setjmp(buf)) goto \*R20 ;*

*goto while\_label ;*

*else7\_label :*

*printf ("You have not created a list!\n\n");*

*goto while\_label ;*

*case8\_label : // case 8*

*if (R18 != 8) goto case9\_label ;*

*if (R8 == R0) goto else8\_label ;*

*R6 = R2 ; // Parameters*

*R7 = R2 ;*

*R20 = (long long)MinimumValue;*

*if(!setjmp(buf)) goto \*R20 ;*

*goto while\_label ;*

*else8\_label :*

*printf ("You have not created a list!\n\n");*

*goto while\_label ;*

*case9\_label : // case 9*

*printf ("You have exited the software!\n\n");*

*return(EXIT\_SUCCESS);*

*else\_label : // default*

*printf ("Please choose one of the following options.\n\n");*

*goto while\_label ;*

*}*

1. ***Υλοποίηση σε Assembly***

.data

.globl array

.globl Input

.globl Inst1

.globl Inst2

.globl Inst3

.globl Inst4

.globl Inst5

.globl Inst6

.globl Inst7

.globl Inst8

.globl Inst9

.globl err1

.globl err2

.globl err3

.globl ttl

.globl addr

.globl crMsg

.globl val

.globl id

.globl nline

.globl addfunc

.globl headadd

.globl err\_notF

.globl errPos

.globl min1

.globl min2

.globl min3

min1: .asciiz "The address of the lowest value is:"

min2: .asciiz " The id of the lowest value is:"

min3: .asciiz " The lowest value is: "

headadd: .asciiz "The address of the list is: "

addfunc: .asciiz "The address of the variable is: "

nline: .asciiz "\n"

id: .asciiz "Id is : "

val: .asciiz "Value is : "

crMsg: .asciiz " values will be inserted.\n"

addr: .asciiz "Enter which value's address you want printed. (1 for the first and so on...)\n"

Input: .asciiz "Welcome. Enter a number from 1 to 9.\n"

Inst1: .asciiz "\nPress 1 to create a list with a specific number of values.\n"

Inst2: .asciiz "Press 2 to insert a value at the end of the list.\n"

Inst3: .asciiz "Press 3 to delete the last value of the list.\n"

Inst4: .asciiz "Press 4 to print a value of the list.\n"

Inst5: .asciiz "Press 5 to print the total number of nodes.\n"

Inst6: .asciiz "Press 6 to print the address of a node.\n"

Inst7: .asciiz "Press 7 to print the address of the list.\n"

Inst8: .asciiz "Press 8 to print the minimum value of the list.\n"

Inst9: .asciiz "Press 9 to exit the software.\n\n"

err1: .asciiz "Invalid input. Choose among 1 and 9.\n"

crList: .asciiz "Enter the number of nodes you want the list to have.\n"

insval: .asciiz "Enter the value to be inserted.\n"

print1: .asciiz "Enter which element you want printed. (1 for the first and so on...)\n"

ttl: .asciiz "The total number of values of the list is: "

err2: .asciiz "A list has already been created.\n"

err3: .asciiz "You need to create a list first.\n"

err\_notF : .asciiz "The requested node does not exist. \n"

errPos : .asciiz "Please enter a positive integer.\n"

array: .align 2

.space 800

.text

.globl main

main:

la $s1, array

li $s2, 0 # the counter.

li $s3, 1 # id

move $t1, $s1

menu:

li $v0, 4

la $a0, Input

syscall

li $v0, 4

la $a0, Inst1

syscall

li $v0, 4

la $a0, Inst2

syscall

li $v0, 4

la $a0, Inst3

syscall

li $v0, 4

la $a0, Inst4

syscall

li $v0, 4

la $a0, Inst5

syscall

li $v0, 4

la $a0, Inst6

syscall

li $v0, 4

la $a0, Inst7

syscall

li $v0, 4

la $a0, Inst8

syscall

li $v0, 4

la $a0, Inst9

syscall

li $v0, 5

syscall

move $s0, $v0

blez $s0, err\_label # The program checks if the user has given an input less than one...

bgt $s0, 9, err\_label # ...or more than 9.

######################################################################################

bne $s0, 1, case2 #the program will skip the code below if the user has not chosen one.

bne $s2, 0, err\_l2

li $v0, 4

la $a0, crList

syscall

li $v0, 5

syscall

move $s0, $v0

add $s2, $s2, $s0

move $a0, $s0 # the number of values to be inserted.

move $a1, $s1 # address of head.

move $a2, $s2 # counter.

move $a3, $s3 # id.

jal CreateList

move $s3 , $v0

addi $s3 , $s3 , -1

b menu

######################################################################################

case2:

bne $s0, 2, case3

beq $s2, 0, err\_l3

li $v0, 4

la $a0, insval

syscall

li $v0, 5

syscall

move $s0, $v0

addi $s2, $s2, 1

move $a0, $s0 # The value to be inserted.

move $a1, $s1 # Address of head.

move $a2, $s2 # The counter.

move $a3, $s3 # The id.

jal insertNode

move $s3 , $v0

b menu

######################################################################################

case3:

bne $s0, 3, case4

beq $s2, 0, err\_l3

move $a1, $s1 #Address of head.

move $a2, $s2 #The counter.

jal deleteLastNode

move $s2, $v0

b menu

#######################################################################################

case4:

bne $s0, 4, case5

beq $s2, 0, err\_l3

li $v0, 4

la $a0, print1

syscall

li $v0, 5

syscall

move $s0, $v0

blez $s0, errPositive # The program checks if the user has given an input less than one...

bgt $s0, $s2, err\_notFound # ...or a value bigger than the length of the list.

move $a0, $s0 # the value (its id) that we want printed.

move $a1, $s1 # Address of Head.

move $a2, $s2 # The counter.

jal PrintValue

b menu

#######################################################################################

case5:

bne $s0, 5, case6

beq $s2, 0, err\_l3

move $a2 , $s2

jal NumberOfNodes

b menu

#######################################################################################

case6:

bne $s0, 6, case7

beq $s2, 0, err\_l3

li $v0, 4

la $a0, addr

syscall

li $v0, 5

syscall

blez $v0, errPositive # The program checks if the user has given an input less than one...

bgt $v0, $s2, err\_notFound # ...or a value bigger than the length of the list.

move $a0, $v0

jal PrintValueAddress

b menu

#######################################################################################

case7:

bne $s0, 7, case8

beq $s2, 0, err\_l3

jal PrintListAddress

li $v0, 4

la $a0, nline

syscall

b menu

#######################################################################################

case8:

bne $s0, 8, case9

beq $s2, 0, err\_l3

move $a1, $s1 #Address of Head.

move $a2, $s2 #Counter.

jal FindMinValue

b menu

#######################################################################################

case9:

li $v0, 10

syscall

#######################################################################################

err\_label:

li $v0, 4

la $a0, err1

syscall

b menu

##########################################################################################

err\_l2:

li $v0, 4

la $a0, err2

syscall

b menu

##########################################################################################

err\_l3:

li $v0, 4

la $a0, err3

syscall

b menu

##########################################################################################

err\_notFound:

li $v0, 4

la $a0, err\_notF

syscall

b menu

##########################################################################################

errPositive:

li $v0, 4

la $a0, errPos

syscall

b menu

##########################################################################################

#main has ended

##########################################################################################

CreateList:

move $t0, $a0 #number of elements to be added.

move $t1, $a1 #address of the array.

move $t2, $a2 #counter

move $t3, $a3 #id.

li $v0, 1

move $a0, $t0

syscall

# The 3 lines above print the number of values that are going to be inserted.

li $v0, 4

la $a0, crMsg

syscall

# The message itself

inputloop:

li $v0, 4

la $a0, insval

syscall

#Print string

li $v0, 5

syscall

#Read integer.

move $t4, $v0 #the value

sw $t3, 0($t1)

addi $t1, $t1, 4

sw $t4, 0($t1)

addi $t1, $t1, 4

addi $t0, $t0, -1

addi $t3, $t3, 1

move $v0 , $t3

beq $t0, 0, exitfunc

b inputloop

exitfunc:

jr $ra

##########################################################################################

insertNode:

move $t0, $a0 #value to be inserted

move $t1, $a1 #address of the array

move $t2, $a2 #counter

move $t4, $a3 #counter

addi $t4, $t4, 1 #increase id.

li $t3, 1 #variable to tranverse the list.

ins\_loop:

addi $t1, $t1, 8

addi $t3, $t3, 1

bne $t3, $t2, ins\_loop # branch

inst:

sw $t4, 0($t1)

addi $t1, $t1, 4

sw $t0, 0($t1)

move $v0 , $t4

jr $ra

##########################################################################################

deleteLastNode:

move $t1, $a1 #address of the array

move $t2, $a2 #counter

li $t3, 1 # variable to tranverse the list.

trav3:

beq $t3, $t2, del

addi $t3, $t3, 1

addi $t1, $t1, 8

b trav3

del:

addi $t2, $t2, -1

move $v0, $t2

jr $ra

##########################################################################################

PrintValue:

move $t0, $a0 #value to be printed

move $t1, $a1 #address of the array

move $t2, $a2 #counter

li $t3, 1 # variable to tranverse the list.

trav:

beq $t3, $t0, PrintVal

addi $t3, $t3, 1

addi $t1, $t1, 8

b trav

PrintVal:

lw $t4, 0($t1)

addi $t1, $t1, 4

lw $t5, 0($t1)

addi $t1, $t1, 4

li $v0, 4

la $a0, id

syscall

li $v0, 1

move $a0, $t4

syscall

li $v0, 4

la $a0, nline

syscall

li $v0, 4

la $a0, val

syscall

li $v0, 1

move $a0, $t5

syscall

li $v0, 4

la $a0, nline

syscall

jr $ra

#############################################################################################

PrintValueAddress:

move $t0, $a0 #value to be printed

move $t1, $a1 #address of the array

move $t2, $a2 #counter

li $t3, 1 # variable to tranverse the list.

trav2:

beq $t3, $t0, PrintValAdd

addi $t3, $t3, 1

addi $t1, $t1, 8

b trav2

PrintValAdd:

la $t4, 0($t1)

li $v0, 4

la $a0, addfunc

syscall

li $v0, 1

move $a0, $t4

syscall

li $v0, 4

la $a0, nline

syscall

jr $ra

#############################################################################################

PrintListAddress:

move $t1, $a1 #address of the array

li $v0, 4

la $a0, headadd

syscall

li $v0, 1

move $a0, $t1

syscall

jr $ra

#############################################################################################

FindMinValue:

move $t1, $a1 #Address of Head.

move $t2, $a2 #Counter (number of values).

addi $t2 , $t2 , -1 # The comparisons that must happen are counter - 1

lw $t3, 0($t1) # Holds the id of the lowest Value.

move $t4, $t1 # Holds the lowest value's address.

addi $t1, $t1, 4

lw $t6, 0($t1) # Holds the lowest value.

beq $t2 , 0 , OneElement # The case where there is only one element

addi $t1, $t1, 8

MinLoop:

lw $t5, 0($t1)

blt $t5, $t6, newMin

addi $t1, $t1, 8

addi $t2, $t2, -1

bne $t2, $zero, MinLoop

b endfunc

newMin:

move $t6, $t5 # Insert the new min value.

addi $t1, $t1, -4

lw $t7, 0($t1)

move $t3, $t7

move $t4, $t1 # Insert the new address.

addi $t2, $t2, -1

addi $t1, $t1, 12

bne $t2, $zero, MinLoop

endfunc:

li $v0, 4

la $a0, min1

syscall

li $v0, 1

move $a0, $t4

syscall

li $v0, 4

la $a0, nline

syscall

li $v0, 4

la $a0, min2

syscall

li $v0, 1

move $a0, $t3

syscall

li $v0, 4

la $a0, nline

syscall

li $v0, 4

la $a0, min3

syscall

li $v0, 1

move $a0, $t6

syscall

li $v0, 4

la $a0, nline

syscall

b endfunc2

################

OneElement :

li $v0, 4

la $a0, min1

syscall

li $v0, 1

move $a0, $t4

syscall

li $v0, 4

la $a0, nline

syscall

li $v0, 4

la $a0, min2

syscall

li $v0, 1

move $a0, $t3

syscall

li $v0, 4

la $a0, nline

syscall

li $v0, 4

la $a0, min3

syscall

li $v0, 1

move $a0, $t6

syscall

li $v0, 4

la $a0, nline

syscall

endfunc2 :

jr $ra

##########################################################################################

NumberOfNodes :

move $t1,$a2

li $v0, 4

la $a0, ttl

syscall

li $v0, 1

move $a0, $t1

syscall

li $v0, 4

la $a0, nline

syscall

jr $ra