23/11/2015

Αναφορά 4ου εργαστηρίου

<u>Προεργασία</u>

Στο 4ο εργαστήριο μας ζητήθηκε ,αρχικά, να τροποποιήσουμε το πρόγραμμα σε Clang του προηγούμενου εργαστηρίου .Η τροποποίηση αυτή αφορούσε πρώτον τη συνάρτηση δημιουργίας λίστας ,στην όποια ο χρήστης έδινε τον αριθμό εισαγωγής κόμβων στη λίστα ,καθώς και δεύτερον τη συνάρτηση διαγραφής κόμβου , η όποια μετά την τροποποίηση θα έπρεπε να διαγράφει τον τελευταίο κόμβο της λίστας αντί του πρώτου. Επιπλέον , μας ζητήθηκε να προσθέσουμε στο πρόγραμμα μας μια νέα συνάρτηση , η οποία θα εκτυπώνει το στοιχείο της λίστας με το μικρότερο value. Στο δεύτερο μέρος του εργαστηρίου , χρειάστηκε να μετατρέψουμε το πρόγραμμα από γλώσσα Clang σε γλώσσα Assembly με κάποιες αλλαγές , όπως το γεγονός ότι τα δεδομένα θα πρέπει να είναι στατικά (στην Assembly) και ότι τα δεδομένα από το χρήστη θα δίνονται μέσα στην συνάρτηση main, πριν την κλήση της κατάλληλης συνάρτησης.

<u>Περιγραφή ζητούμενων</u>

Ο σκοπός της άσκησης είναι η περαιτέρω εξοικείωση με τη γλώσσα Clang, μέσω της πλήρους κατανόησης της λειτουργίας των καταχωρητών και των εντολών όπως η goto, η longjmp, καθώς και τη setjmp. Επιπροσθέτως, έγινε αντιληπτό ότι η υλοποίηση και η πλήρης λειτουργικότητα των ζητούμενων συναρτήσεων σε κώδικα Clang θα είναι καθοριστική για την κατασκευή του προγράμματος σε Assembly.

Όσον αφορά την Assembly, σκοπός είναι να κατανοήσουμε την κλήση συναρτήσεων με την χρήση των εντολών jal και jr καθώς και να μάθουμε να τηρούμε τις συμβάσεις χρήσης των καταχωρητών (για την κλήση συναρτήσεων) .

<u>Περιγραφή εκτέλεσης</u>

Αρχικά , σε κώδικα C ,τροποποιήσαμε την συνάρτηση createList() του 3ου εργαστηρίου , προσθέτοντας ένα while loop ,με επαναλήψεις τόσες όσες τα στοιχεία που θέλει να εισάγει ο χρήστης , και κάνοντας για κάθε στοιχείο malloc. Έπειτα, κάναμε την μετατροπή της συνάρτησης σε γλώσσα Clang , όπως ζητείται από το εργαστήριο :

Clang : createList(πριν τη μετατροπή)	Clang : createList(μετά τη μετατροπή)

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

}
```

```
void createList(){
 R22 = 0;
 while_label_1:
if(R22 >= R7) goto after_loop_1;
printf("\nInsert node %d of
%d\n",R22+1,R7);
struct list *node, *returnValue;
  struct list *node2;
  R20 = (int)node;
node2=(struct list*)R4;
printf("Give value: ");
  scanf("%hd",&R6);
   printf("Give ID: ");
scanf("%d",&R5);

R20 = (int)malloc(sizeof(struct list));
((struct list*)R20)->value = R6;
 ((struct list*)R20)->id = R5;
 ((struct list*)R20)->next = NULL;


   if(R4 != (int)NULL) goto after_cond;
   R4 = R20;
  R21 = R20;
   ((struct list*)R21)->next = NULL;
R22 = R22 + 1;
goto while_label_1;
after_cond:
while_label:
if(((struct list*)R21)->next == NULL) goto
after_loop;
  R21 = (int)((struct list*)R21)->next;
goto while_label;
   after_loop:

  ((struct list*)R21)->next = (struct list*)R20;
  R22 = R22 + 1;
  goto while_label_1;
 after_loop_1:
            R2 = R4;
            JR31;}
```

Την ίδια διαδικασία κάναμε και για να υλοποιήσουμε την συνάρτηση διαγραφής του τελευταίου κόμβου. Στην ουσία , η συνάρτηση αυτή που υλοποιήσαμε, διασχίζει τη λίστα μέχρι τον προτελευταίο κόμβο. Όποτε φτάνουμε στον προτελευταίο κόμβο, μηδενίζουμε το περιεχόμενο του τελευταίου και τον αποδεσμεύουμε (free) ,κάνοντας τον ελεύθερο για το λειτουργικό σύστημα , όπως φαίνεται παρακάτω :

Τμήμα της συνάρτησης deleteNodeList() σε Clang

```
if(!(R22 >= 2)) goto after_if; // έλεγχος, έτσι ώστε ο αριθμός των κόμβων >2
        while_label:
        if(((struct list *)R21)->next->next == NULL)goto after_loop;//σκοπός είναι να
φτάσουμε έναν κόμβο μετά τον τελευταίο για να αποδεσμέυσουμε τον τελευταίο
        R21 = (int)((struct list*)R21)->next;
        goto while_label;
        after_loop:

    free(((struct list*)R21)->next);
   ((struct list*)R21)->next = NULL;
```

Για την υλοποίηση της συνάρτησης printMinValue() , χρειάστηκε να κάνουμε διάσχιση της λίστας (σε γλώσσα C) και να κρατήσουμε σε προσωρινό δείκτη, όποια τιμή(value) κόμβου ήταν μικρότερη της τιμής των προηγούμενων κόμβων. Έτσι , όταν τέλειωνε η διάσχιση της λίστας ,είχαμε τον κόμβο με την μικρότερη τιμή (value). Μετά μετατρέψαμε την συνάρτηση σε Clang και την ενσωματώσαμε στο πρόγραμμα μας ,όπως φαίνεται στο παράρτημα.

Όσον αφορά την Assembly ,ξεκινήσαμε δημιουργώντας έναν πινάκα για 100 κόμβους :

```
array: .align 2
       .space 800     # space for 100 nodes

         #800=((4(int value)+4(int id))*100)
```

Στη συνεχεία υλοποιήσαμε όλες τις συναρτήσεις της Clang με τη σειρά , χρησιμοποιώντας τις εντολές branch οπού υπήρχαν συνθήκες (σε if και loops), και τις εντολή j (jump) για την υλοποίηση των ίδιων των βρόγχων , όπως φαίνεται παρακάτω :

Τμήμα της συνάρτησης insertArray()

```
while_label:        # the perpose of this loop is to
go the head to the last element

bge $t4, $t6, after_loop

add $t1, $t1, 8              # move head to the
next cellar
addi $t4, $t4, 1           # add 1 to counter

j while_label

after_loop:  ….
```

<u>Συμπεράσματα</u>

Εκτελώντας την άσκηση κατανοήσαμε την χρήση των καταχωριστών για την εκτέλεση διαφόρων λειτουργιών σε ένα πρόγραμμα , μέσω της γλώσσας Clang και έπειτα μέσω της Assembly. Γενικότερα εμβαθύναμε παραπάνω στην Clang , η όποια με τη σειρά της

βοήθησε να γράψουμε τον κώδικα σε Assembly, αφού οι συνθήκες , οι κλήσεις των συναρτήσεων και η δομή των βρόγχων ήταν ,εν τελεί, κοινά στοιχεία των δυο προγραμμάτων. Επιπλέον , συμπεράναμε ότι η πιστή τήρηση των συμβάσεων επιτρέπει σε μας να δουλεύουμε ταυτοχρόνως διαφορετικά κομμάτια κώδικα του ίδιου προγράμματος, χωρίς να δημιουργείται κανένα λειτουργικό πρόβλημα (στον κώδικα) και στην συνέχεια να τα ενσώματώνουμε στην main.

Παράρτημα

Κώδικας 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;

};


void menuPrint();

void createList();

void insertNodeList();

void deleteNodeList();
```

```
void printNode();

void printNumOfNodes();

void printSpecialNode();

void printAddressList();

void printMinValue();

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){

        int ch;

        short v;


        void(*foo)(void);

   foo=&menuPrint;

   void(*foo1)(void);

   foo1=&createList;

   void(*foo2)(void);

   foo2=&insertNodeList;

   void(*foo3)(void);

   foo3=&deleteNodeList;

   void(*foo4)(void);

   foo4=&printNode;
```

```
void(*foo5)(void);
foo5=&printNumOfNodes;
void(*foo6)(void);
foo6=&printSpecialNode;
void(*foo7)(void);
        foo7=&printAddressList;
        void(*foo8)(void);
        foo8=&printMinValue;

do_label:

                R15=(int)foo;
        jal(R15);
        if(R2 != 1) goto else_label_1;
                printf("\nGive the number of nodes that you want: ");
                scanf("%d",&ch);
                R7 = ch;
                R16=(int)foo1;
                jal(R16);
                printf("\n\n----PrintAll funcion----\n\n");
                printAll((struct list*)R4,1);
                printf("\n----PrintAll funcion----\n\n");

                goto after_cond;
                else_label_1:
                if(R2 != 2) goto else_label_2;
                        printf("Give value: ");
```

```
scanf("%hd",&v);
R6 = (int)v;
printf("Give ID: ");
scanf("%d",&R5);
R17=(int)foo2;
jal(R17);
                    printf("\n\n----PrintAll funcion----\n\n");
printAll((struct list*)R4,1);
printf("\n----PrintAll funcion----\n\n");
goto after_cond;
else_label_2:
if(R2 != 3) goto else_label_3;
        R20=(int)foo5;
        jal(R20);
        R18 = (int)foo3;
        jal(R18);
        printf("\n\n----PrintAll funcion----\n\n");
        printAll((struct list*)R4,1);
        printf("\n----PrintAll funcion----\n\n");
goto after_cond;
else_label_3:
if(R2 != 4) goto else_label_4;
        R19=(int)foo4;
        jal(R19);
        goto after_cond;
else_label_4:
                    if(R2 !=5)goto else_label_5;
```

```
                    R20=(int)foo5;
                    jal(R20);
                    printf("\nThe number of nodes is: %d\n",R2);
                goto after_cond;
                else_label_5:
        if(R2 != 6) goto else_label_6;
                R21=(int)foo6;
                jal(R21);
                goto after_cond;
                else_label_6:
        if(R2 != 7) goto else_label_7;
                R22=(int)foo7;
                jal(R22);
                goto after_cond;
                else_label_7:
                if(R2 != 8) goto after_cond;
                R24=(int)foo8;
                jal(R24);
                goto after_cond;
        after_cond:
                if(R2 == 9) goto after_loop;
    goto do_label;
    after_loop:
    system("pause");
    printf("\n\n\n\n----------Bye!----------\n\n");
    return 0;
}
```

```
void menuPrint(){
        int R8;
        printf("-----------------Menu-----------------\n\n");
        printf("\nType (1) to create list\n");
        printf("Type (2) to insert a node\n");
        printf("Type (3) to delete the last 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 the address of special item section\n");
        printf("Type (7) to print the address of list\n");
        printf("Type (8) to print the minimum value of all nodes\n");
        printf("Type (9) to exit\nYour choice: ");
        scanf("%d",&R8);
        printf("\n-----------------Menu-----------------\n\n");
        R2 = R8;
        JR31;
}

void createList(){
         R22 = 0;
         while_label_1:
                 if(R22 >= R7) goto after_loop_1;

                         printf("\nInsert node %d of %d\n",R22+1,R7);
                         struct list *node, *returnValue;
```

```
        struct list *node2;
        R20 = (int)node;

        node2=(struct list*)R4;

        printf("Give value: ");
scanf("%hd",&R6);
printf("Give ID: ");
        scanf("%d",&R5);

        R20 = (int)malloc(sizeof(struct list));

        ((struct list*)R20)->value = R6;
        ((struct list*)R20)->id = R5;
        ((struct list*)R20)->next = NULL;


        if(R4 != (int)NULL) goto after_cond;
                R4 = R20;
                R21 = R20;
                ((struct list*)R21)->next = NULL;
                R22 = R22 + 1;
                goto while_label_1;

        after_cond:

        while_label:
```

```
                    if(((struct list*)R21)->next == NULL) goto after_loop;
                R21 = (int)((struct list*)R21)->next;

            goto while_label;
                after_loop:

                    ((struct list*)R21)->next = (struct list*)R20;

            R22 = R22 + 1;
            goto while_label_1;
            after_loop_1:
            R2 = R4;
            JR31;

}

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

  R20 = (int)node;
  R21 = (int)node2;

  node2=(struct list*)R4;
  R21 = (int)node2;

  R20 = (int)malloc(sizeof(struct list));
```

```
  ((struct list*)R20)->value = R6;
  ((struct list*)R20)->id = R5;
  ((struct list*)R20)->next = NULL;


  if(R4 != (int)NULL) goto after_cond;
   returnValue = (struct list*)R20;
   R4 = R20;
          R2 = R4;
   JR31;
  after_cond:

  while_label:
          if(((struct list*)R21)->next == NULL) goto after_loop;
      R21 = (int)((struct list*)R21)->next;
      goto while_label;
   after_loop:

  ((struct list*)R21)->next = (struct list*)R20;
  R2 = R4;
  JR31;
 }

void deleteNodeList(){
          R21 = R4;
          R22 = R28; // Register 28 has the num of nodes from the return value of
"printNumOfNodes" function
```

```
        if(R4 != (int)NULL) goto after_condit;
                printf("\n----Empty List!----\n");
                R2 = R4;
                JR31;
        after_condit:

        if(R22 != 1) goto after_cond;
                R4 = (int)NULL;
                printf("\n----Empty List!----\n");
                R2 = R4;
                JR31;
        after_cond:


        if(!(R22 >= 2)) goto after_if;
                while_label:
                if(((struct list *)R21)->next->next == NULL)goto after_loop;
                R21 = (int)((struct list*)R21)->next;
                goto while_label;
                after_loop:

                free(((struct list*)R21)->next);
        ((struct list*)R21)->next = NULL;
        after_if:

R2 = R4;
```

```
        JR31;

}

void printNode(){

        struct list *node;
        node = (struct list*)R4;
        R24 = (int)node;
        printf("Give ID: ");
        scanf("%d",&R23);
        R22 = 0;
        while_label:
        if(((struct list*)R24) == NULL) goto after_loop;
                if(R23 != ((struct list*)R24)->id) goto after_cond;
                        printf("\nNode info:\nValue: %d\nID: %d\n\n",((struct list*)R24)-
>value,((struct list*)R24)->id);
                        R22 = 1;
                after_cond:
                R24 = (int)((struct list*)R24)->next;
        goto while_label;
        after_loop:
        if(R22 == 1) goto after_if;
                printf("This node does not exist!\n");
        after_if:
        JR31;
}
```

```
void printNumOfNodes(){
        R22 = R4;
        R28=0;

        while_label:
                if((struct list *)R22 == NULL)goto after_loop;
                        R28=R28+1;
                        R22=(int)((struct list*)R22)->next;
                goto while_label;
        after_loop:
        R2=R28; //num of nodes
        JR31;
}

void printSpecialNode(){
        struct list* node;
        char ch;
   node = (struct list*)R4;
   R22  = (int)node;
        R27 = (short)R24;
        printf("Value or ID?[v-i]: ");
        scanf("\n%c",&ch);
        R26 = (int)ch;

                if((char)R26 != 'i')goto else_label;
```

```
                printf("Give id: ");
                scanf("%d",&R25);
                while_label_1:
        if((struct list*)R22 == NULL) goto after_loop_1;
          if(((struct list*)R22)->id != R25) goto after_cond_1;
                printf("\nNode info: \n");
                printf("ID: %d\n",((struct list*)R22)->id);
                                        printf("Value: %d\n",((struct list*)R22)-
>value);
            printf("ID address: %x\n",&((struct list*)R22)->id);
                                        printf("Value address: %x\n",&((struct
list*)R22)->value);
                                after_cond_1:
                                R22 = (int)((struct list*)R22)->next;
                goto while_label_1;
        after_loop_1:
                goto after_cond;
                else_label:
                if ((char)R26 != 'v') goto after_cond;
                        printf("Give value: ");
                        scanf("%d",&R27);
                        while_label_2:
        if(((struct list*)R22) == NULL) goto after_loop_2;
                                if(((struct list*)R22)->value != R27) goto
after_cond_2;
                                        printf("\nNode info: \n");
                                        printf("Value: %d\n",((struct list*)R22)-
>value);
                printf("ID: %d\n",((struct list*)R22)->id);
```

```
                        printf("Value address: %x\n",&((struct list*)R22)->value);
                                                printf("ID address: %x\n",&((struct
list*)R22)->id);
                        after_cond_2:
                        R22 = (int)((struct list*)R22)->next;


                goto while_label_2;
        after_loop_2:
        after_cond:
        R4 = R22;
        R2 = R4;
        JR31;
}

void printAddressList() {
            struct list* node;
            node=(struct list*)R4;
            printf("Address of list: %x\n",R4);
             R2 = R4;
         JR31;
}
void printMinValue(){
            struct list *temp;
        R23 = R4;
        R28=1000;
        while_label_1:
        if(!(((struct list*)R23) != NULL)) goto after_loop_1;
                        if(!(((struct list*)R23)->value < R28)) goto after_if;
```

```
                                R28 = (int)((struct list*)R23)->value;
                                R27 = (int)((struct list*)R23)->id;
                        after_if:
                        R23 = (int)((struct list*)R23)->next;
                goto while_label_1;
                after_loop_1:

                R23 = (int)(struct list*)R4;

                while_label_2:
                if(((struct list*)R23) == NULL) goto after_loop_2;
                        if(R27 !=  ((struct list*)R23)->id) goto after_cond;
                                printf("\n\nMinimum Value: %d\n",R28);
                                printf("Node %d Info:\n",R27);
                                printf("Value: %d \nID: %d\n", ((struct list*)R23)->value,
((struct list*)R23)->id);
                        after_cond:
                R23 = (int)((struct list*)R23)->next;
                goto while_label_2;
                after_loop_2:
        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;
}
```

<u>Κώδικας Assembly</u>

```
.data
.globl main
.globl printmenu
.globl createArray
.globl insertArray
.globl deleteLast
.globl printItem
.globl numOfItems
.globl printAddressItem
.globl printArrayAddress
.globl printMinValue

		# Messages
menu_print:   .asciiz "\n\nType (1) to create the array\nType (2) to insert an item to array\nType (3) to delete the last item of array\nType (4) to print special item\nType (5) to print the number of items\nType (6) to print the address of special item\nType (7) to print the address of array\nType (8) to print the minimum value of all items of array\nType (9) to exit\n"
askint:       .asciiz "\nPlease type your choice: "
```

```
askid:        .asciiz "\nGive id: "
askvalue:     .asciiz "\nGive value: "
bye:          .asciiz "\nProgram terminated. Bye!"
emptyarr:     .asciiz "\n----Empty Array----\n"
size:         .asciiz "\nGive the number of items: "
printinfo:    .asciiz "\nNode info: \n"
valueinfo:    .asciiz "\nValue: "
idinfo:       .asciiz "\nID: "
errornode:    .asciiz "\nThis node does not exist!\n"
idaddress:    .asciiz "\nID address: "
valueaddress: .asciiz "\nValue address: "
arrayaddress: .asciiz "\nAddress of array: "
minvalue:     .asciiz "\nMinimum value: "
number:       .asciiz "\nThe number of items is: "
node:         .asciiz "\nGive the item that you want: "
newline:      .asciiz "\n"

array: .align 2
              .space 800      # space for 100 nodes

.text

main:

la $s1, array          # $s1 is head of the array
li $s5,9               # $s5 is exit register
li $s7, 0              # initialization of $s6 which is the counter of nodes
```

```
move $t1, $s1

loop:

li $s4,1

jal printmenu

move $s0, $v0

case_1:             # case if choice is 1

bne $s0,$s4,case_2

li $v0, 4           # print how items want
la $a0, size
syscall

li $v0,5
syscall

move $s6, $v0               # $s6 has the number of items
move $a3, $s6               # counter as argument to $a3
move $a0, $s1               # head array as argument to $a1

jal createArray
```

```
move $s6, $v0

move $a0, $s1


jal printall


j loop


case_2:              # case if choice is  2


addi $s4,$s4,1


bne $s0,$s4,case_3


li $v0, 4            # message to read value

la $a0, askvalue

syscall


li $v0, 5            # read value

syscall

move $s2, $v0        # move value to $s2


li $v0, 4            # message to read id

la $a0, askid

syscall


li $v0, 5            # read id

syscall
```

```
move $s3, $v0                # move id to $s3

move $a0, $s1        # head array as argument to $a0
move $a1, $s2                # value as argument to $a1
move $a2, $s3        # ID as argument to $a2
move $a3, $s6                # counter as argument to $a3

jal insertArray

move $s6, $v0

li  $v0, 1           # print integer
move $a0, $s6
syscall

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

move $a0, $s1        # head array as argument to $a0
move $a3, $s6                # counter as argument to $a3
jal printall

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

```
j loop

case_3:             # case if choice is 3

addi $s4,$s4,1

bne $s0,$s4,case_4

move $a0, $s1

move $a3, $s6

jal deleteLast

move $s6,$v0

move $a0, $s1       # head array as argument to $a0
move $a3, $s6               # counter as argument to $a3

jal printall

j loop

case_4:             # case if choice is 4

addi $s4,$s4,1
```

```
bne $s0,$s4,case_5

move $a0,$s1

jal printItem

j loop

case_5:             # case if choice is 5

addi $s4,$s4,1

bne $s0,$s4,case_6

move $a3, $s6               # counter as argument to $a3

jal numOfItems

j loop

case_6:             # case if choice is 6

addi $s4,$s4,1

bne $s0,$s4,case_7

move $a0,$s1
```

```
jal printAddressItem

j loop

case_7:             # case if choice is 7

addi $s4,$s4,1
bne $s0,$s4,case_8

move $a0, $s1

jal printArrayAddress

j loop

case_8:             # case if choice is 8

addi $s4,$s4,1
bne $s0,$s4,case_9

move $a0, $s1
move $a3, $s6

jal printMinValue
```

```
j loop

case_9:              # case 9 is the jump to loop label

li  $v0, 4           # print new line
la  $a0, newline
syscall
bne $s0, $s5,loop      # loop condition

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


li $v0, 4            # print string of terminated program
la $a0, bye
syscall

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

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

printmenu:           # menu function

li $v0, 4            # print menu string
la $a0, menu_print
```

```
syscall

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

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

li  $v0, 5          # read choice
syscall


move $a0,$v0
move $v0,$a0
jr $ra                              # return to main

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

createArray:

move $t6, $a3       # choice
move $t1, $a0               # head
li $t5,0            # counter
```

```
loop_1:


bge $t5,$t6,after_loop_1
li $v0,4
la $a0,askvalue       # print message  to ask value
syscall

li $v0,5              # read value
syscall

move $t2, $v0
sw $t2,0($t1)         # move value to t2

li $v0,4
la $a0,askid          # print message ask id
syscall

li $v0,5              # read id
syscall



move $t3, $v0         # move id to $t3
sw $t3,4($t1)

addi $t5,$t5,1        # counter of nodes
addi $t1,$t1,8        # move head address
```

```
bne $t5,$t6,loop_1

after_loop_1:



move $v0, $t6

jr $ra

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

insertArray:

move $t1, $a0                    # $t1 has the head of array
move $t2, $a1                    # $t2 has the value
move $t3, $a2                    # $t3 has the ID
move $t6, $a3                    # $t7 has the number of elements


li $t4,0                         # $t4 is a help counter to the next while loop

while_label:          # the perpose of this loop is to go the head to the last element

bge $t4, $t6, after_loop

add $t1, $t1, 8                  # move head to the next cellar
addi $t4, $t4, 1              # add 1 to counter
```

```
j while_label

after_loop:

sw $t2, 0($t1)                  # store the value and the ID
sw $t3, 4($t1)

addi $t6, $t6, 1            # add 1 to the number of nodes

move $v0, $t6                   # return value is the number of nodes

jr $ra

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

numOfItems:

move $t6,$a3

li $v0, 4           # print the arrat's size
la $a0, number
syscall

li  $v0, 1          # print integer
move $a0, $t6
syscall
```

```
jr $ra

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

printItem:

move $t1,$a0          # head

li $t5,1              # counter
li $t8,0              # boolean flag

li $v0,4
la $a0,node           # print message ask node
syscall

li $v0,5              # read id
syscall

move $t4, $v0

loop_2:

beq $t5,$t4,after_loop_2

addi $t1,$t1,8
```

```
addi $t5,$t5,1

j loop_2

after_loop_2:

lw $t2,0($t1)
lw $t3,4($t1)

addi $t8,$t8,1

li $v0,4
la $a0,valueinfo       # print message
syscall

li  $v0, 1           # print integer
move $a0, $t2
syscall

li $v0,4
la $a0,idinfo         # print message
syscall

li  $v0, 1           # print integer
move $a0, $t3
syscall
```

```
bne $t8,$zero,after_if

li $v0,4
la $a0,errornode      # print message
syscall

after_if:
jr $ra

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

printAddressItem:

move $t1,$a0          # head

li $t5,1              # counter
li $t8,0              # boolean flag

li $v0,4
la $a0,node           # print message ask node
syscall

li $v0,5              # read id
syscall

move $t4, $v0
```

```
loop_3:

beq $t5,$t4,after_loop_3

addi $t1,$t1,8
addi $t5,$t5,1

j loop_3

after_loop_3:

addi $t8,$t8,1

li $v0,4
la $a0,valueaddress    # print message
syscall

li  $v0, 1           # print integer
move $a0,$t1
syscall

addi $t1,$t1,4

li $v0,4
la $a0,idaddress       # print message
syscall
```

```
li  $v0, 1           # print integer
move $a0, $t1
syscall

bne $t8,$zero,after_if_3

li $v0,4
la $a0,errornode      # print message
syscall



after_if_3:
jr $ra

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

deleteLast:

move $t1,$a0

move $t6,$a3

li $t5,0              # help counter

bne $t6, $zero, after_cond_del     # case if array is empty
```

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

li $v0, 4               # print string when array is empty
la $a0, emptyarr
syscall

move $v0, $t6

jr $ra

after_cond_del:

loop_4:

beq $t5,$t6,after_loop_4  # the purpose of this loop is to reach to the next of the last item

addi $t1,$t1,8
addi $t5,$t5,1

j loop_4

after_loop_4:
addi $t6,$t6,-1                     # add -1 to the number of items
move $s6,$t6
```

```
lw $t2,-8($t1)
lw $t3,-4($t1)

move $t2,$zero
move $t3,$zero

sw $t2,-8($t1)
sw $t3,-4($t1)

move $v0, $t6

jr $ra

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

printall:
move $t1,$a0
move $t6,$a3
li $t7,0

loop_5:
beq $t7,$t6,after_loop_5
lw $t2,0($t1)
lw $t3,4($t1)

li $v0,4
la $a0,valueinfo       # print message
```

```
syscall

li  $v0, 1           # print integer
move $a0, $t2
syscall

li $v0,4
la $a0,idinfo         # print message
syscall

li  $v0, 1           # print integer
move $a0, $t3
syscall

add $t1,$t1,8
addi $t7,$t7,1

j loop_5

after_loop_5:

jr $ra

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

printArrayAddress:
```

```
move $t1, $a0

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

li $v0, 4           # print message
la $a0, arrayaddress
syscall

li  $v0, 1          # print array address
move $a0, $t1
syscall

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

jr $ra

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

printMinValue:

move $t1, $a0
move $t6, $a3
```

```
move $t5,$zero          # help counter

li $t4, 100             # $t4 holds the minimum value

while_label_min:

bge $t5, $t6, after_loop_min

lw $t2, 0($t1)          # load the value

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

bge $t2,$t4,after_cond_min

move $t4, $t2           # update minimum value

after_cond_min:

addi $t5, $t5, 1        # add 1 to help counter
addi $t1, $t1, 8        # add 8 to head address

j while_label_min

after_loop_min:
```

```
li $v0, 4              # print message
la $a0, minvalue
syscall

li $v0, 1
move $a0, $t4          # print minimum value
syscall

jr $ra

##############################################################################
```