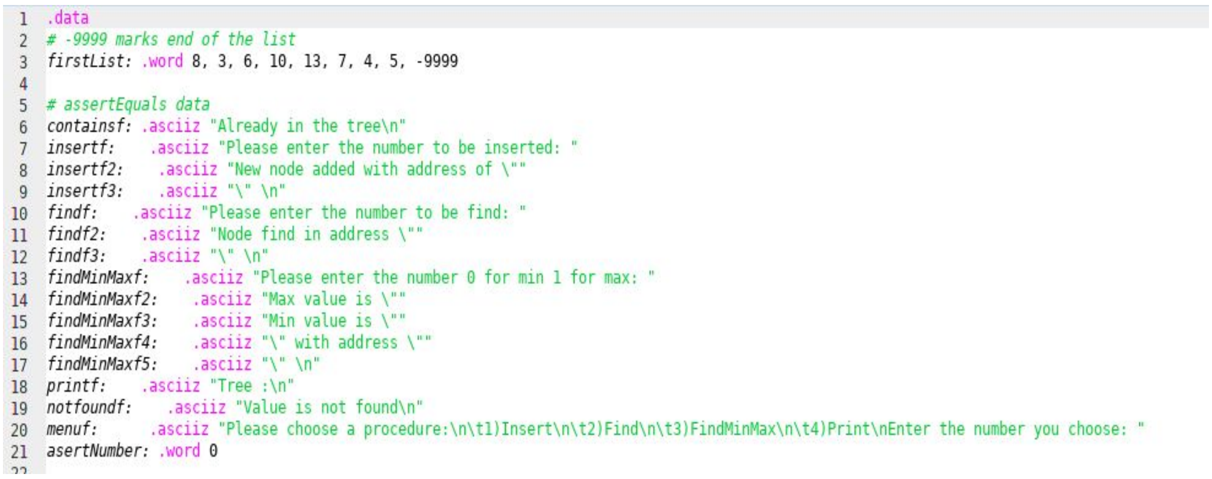
Name: Zekihan AZMAN
No: 250201007

250201007_P1

This is a basic implementation of binary search tree in MIPS assembly language.
Developing and testing done through SPIM simulator [1].
For input and output I followed the file uploaded to cms not the homework pdf.

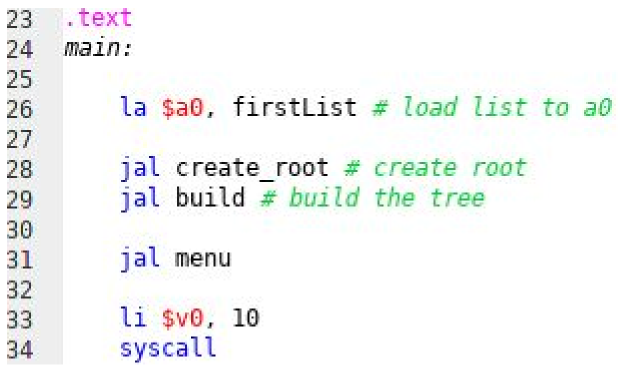
.data area:



```
.data
# -9999 marks end of the list
firstList: .word 8, 3, 6, 10, 13, 7, 4, 5, -9999

# assertEquals data
containsf: .asciiz "Already in the tree\n"
insertf:    .asciiz "Please enter the number to be inserted: "
insertf2:    .asciiz "New node added with address of \""
insertf3:    .asciiz "\" \n"
findf:    .asciiz "Please enter the number to be find: "
findf2:    .asciiz "Node find in address \""
findf3:    .asciiz "\" \n"
findMinMaxf:    .asciiz "Please enter the number 0 for min 1 for max: "
findMinMaxf2:    .asciiz "Max value is \""
findMinMaxf3:    .asciiz "Min value is \""
findMinMaxf4:    .asciiz "\" with address \""
findMinMaxf5:    .asciiz "\" \n"
printf:    .asciiz "Tree :\n"
notfoundf:    .asciiz "Value is not found\n"
menuf:      .asciiz "Please choose a procedure:\n\t1)Insert\n\t2)Find\n\t3)FindMinMax\n\t4)Print\nEnter the number you choose: "
asertNumber: .word 0
```

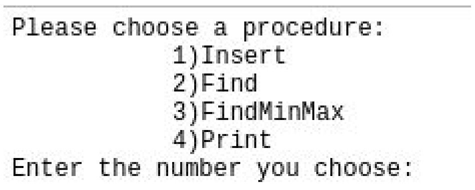
This area contains the first list that used for building bst and all ascii lines for menu.

.text area:



```
.text
main:

    la $a0, firstList # load list to a0

    jal create_root # create root
    jal build # build the tree

    jal menu

    li $v0, 10
    syscall
```

Main method creates a root then builds a tree from first list.
Then calls for menu.
This is the basic menu layout:

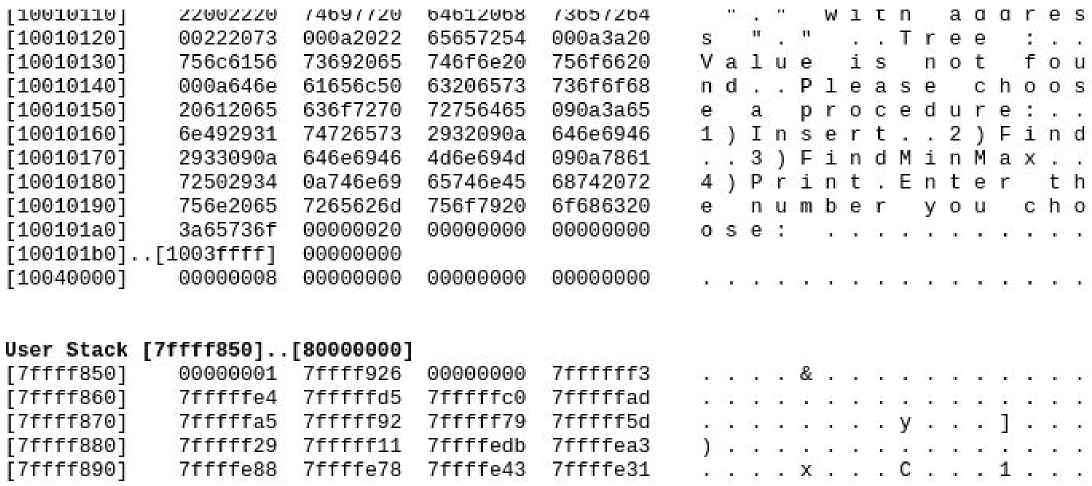


```
Please choose a procedure:
            1)Insert
            2)Find
            3)FindMinMax
            4)Print
Enter the number you choose:
```

Name: Zekihan AZMAN
No: 250201007

create root:

It makes space for 4 integers and give first place to first element of list.

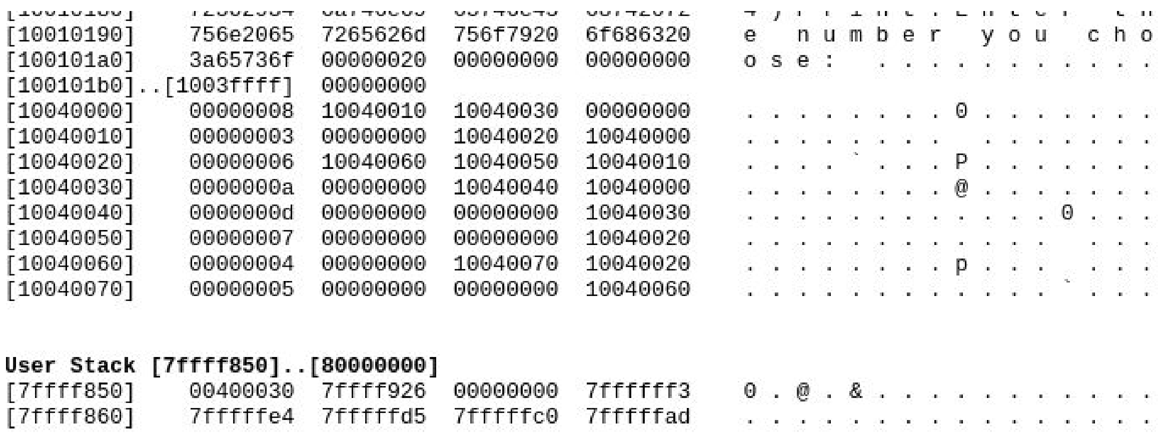


```
[10010110]    22002220  74697720  64612068  73657264     " . "   w i t h   a d d r e s
[10010120]    00222073  000a2022  65657254  000a3a20    s   " . "   . . T r e e   : . .
[10010130]    756c6156  73692065  746f6e20  756f6620    V a l u e   i s   n o t   f o u
[10010140]    000a646e  61656c50  63206573  736f6f68    n d . . P l e a s e   c h o o s
[10010150]    20612065  636f7270  72756465  090a3a65    e   a   p r o c e d u r e : . .
[10010160]    6e492931  74726573  2932090a  646e6946    1 ) I n s e r t . . 2 ) F i n d
[10010170]    2933090a  646e6946  4d6e694d  090a7861    . . 3 ) F i n d M i n M a x . .
[10010180]    72502934  0a746e69  65746e45  68742072    4 ) P r i n t . E n t e r   t h
[10010190]    756e2065  7265626d  756f7920  6f686320    e   n u m b e r   y o u   c h o
[100101a0]    3a65736f  00000020  00000000  00000000    o s e :   . . . . . . . . . . .
[100101b0]..[1003ffff]  00000000
[10040000]    00000008  00000000  00000000  00000000    . . . . . . . . . . . . . . . .


User Stack [7ffff850]..[80000000]
[7ffff850]    00000001  7ffff926  00000000  7ffffff3    . . . . & . . . . . . . . . . .
[7ffff860]    7fffffe4  7fffffd5  7fffffc0  7fffffad    . . . . . . . . . . . . . . . .
[7ffff870]    7fffffa5  7fffff92  7fffff79  7fffff5d    . . . . . . . . . y . . . ] . . .
[7ffff880]    7fffff29  7fffff11  7ffffedb  7ffffea3    ) . . . . . . . . . . . . . . .
[7ffff890]    7ffffe88  7ffffe78  7ffffe43  7ffffe31    . . . . x . . . C . . . 1 . . .
```

As you can see value 8 is in address 10040000 and following 3 bytes are 0.

build:

It goes into a while loop and inserts elements from list to bst until -9999 cames as element.



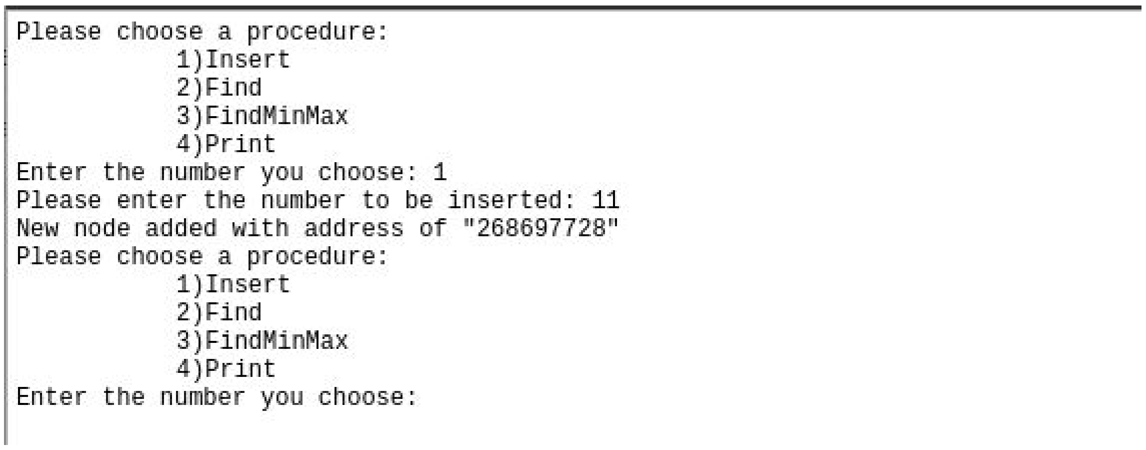
```
[10010180]    72502934  0a746e69  65746e45  68742072    4 ) P r i n t . E n t e r   t h
[10010190]    756e2065  7265626d  756f7920  6f686320    e   n u m b e r   y o u   c h o
[100101a0]    3a65736f  00000020  00000000  00000000    o s e :   . . . . . . . . . . .
[100101b0]..[1003ffff]  00000000
[10040000]    00000008  10040010  10040030  00000000    . . . . . . . . . 0 . . . . . .
[10040010]    00000003  00000000  10040020  10040000    . . . . . . . .   . . . . . . .
[10040020]    00000006  10040060  10040050  10040010    . . . . ` . . . P . . . . . . .
[10040030]    0000000a  00000000  10040040  10040000    . . . . . . . . @ . . . . . . .
[10040040]    0000000d  00000000  00000000  10040030    . . . . . . . . . . . . 0 . . .
[10040050]    00000007  00000000  00000000  10040020    . . . . . . . . . . . .   . . .
[10040060]    00000004  00000000  10040070  10040020    . . . . . . . . p . . . . . . .
[10040070]    00000005  00000000  00000000  10040060    . . . . . . . . . . . . ` . . .


User Stack [7ffff850]..[80000000]
[7ffff850]    00400030  7ffff926  00000000  7ffffff3    0 . @ . & . . . . . . . . . . .
[7ffff860]    7fffffe4  7fffffd5  7fffffc0  7fffffad    . . . . . . . . . . . . . . . .
```

All items of the list is inserted into tree is stored in this memory addresses.

Name: Zekihan AZMAN
No: 250201007

insert:

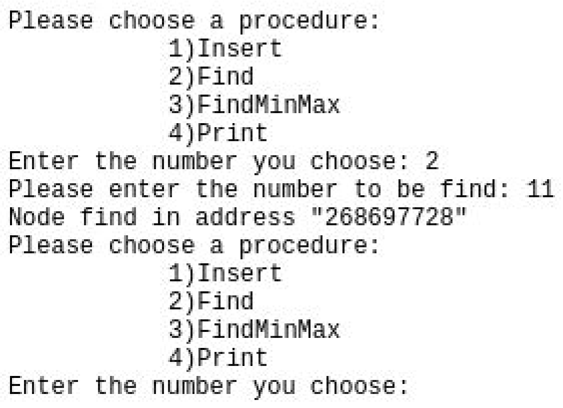
It takes a integer and tries to find its correct spot by going left if smaller ,going right if bigger if it is same with one of the nodes then it is not inserted.



```
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose: 1
Please enter the number to be inserted: 11
New node added with address of "268697728"
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose:
```

find:

Similar to the insert it traverses through tree trying to find a element until it reaches a zero child which means no child of node or it finds its element.

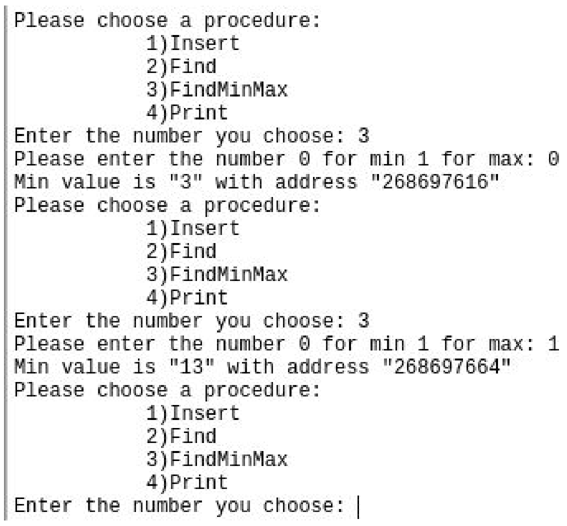


```
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose: 2
Please enter the number to be find: 11
Node find in address "268697728"
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose:
```

Name: Zekihan AZMAN
No: 250201007

findMinMax:

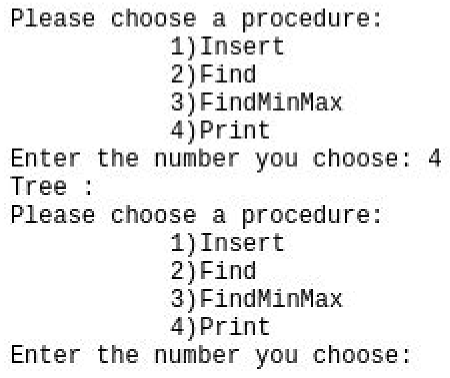
Takes one argument to go minimum or maximum.
When going minimum it goes leftmost child and it's the smallest element possible in the tree.
When going maximum it goes rightmost child and it's the biggest element possible in tree.



```
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose: 3
Please enter the number 0 for min 1 for max: 0
Min value is "3" with address "268697616"
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose: 3
Please enter the number 0 for min 1 for max: 1
Min value is "13" with address "268697664"
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose: |
```

print:

I couldn't be able to implement print method. As it required breadth-first approach and need me to implement a queue and termination of loop is another big issue.



```
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose: 4
Tree :
Please choose a procedure:
           1)Insert
           2)Find
           3)FindMinMax
           4)Print
Enter the number you choose:
```

In menu context it is ready unfortunately i cannot implement print method so there is no output.