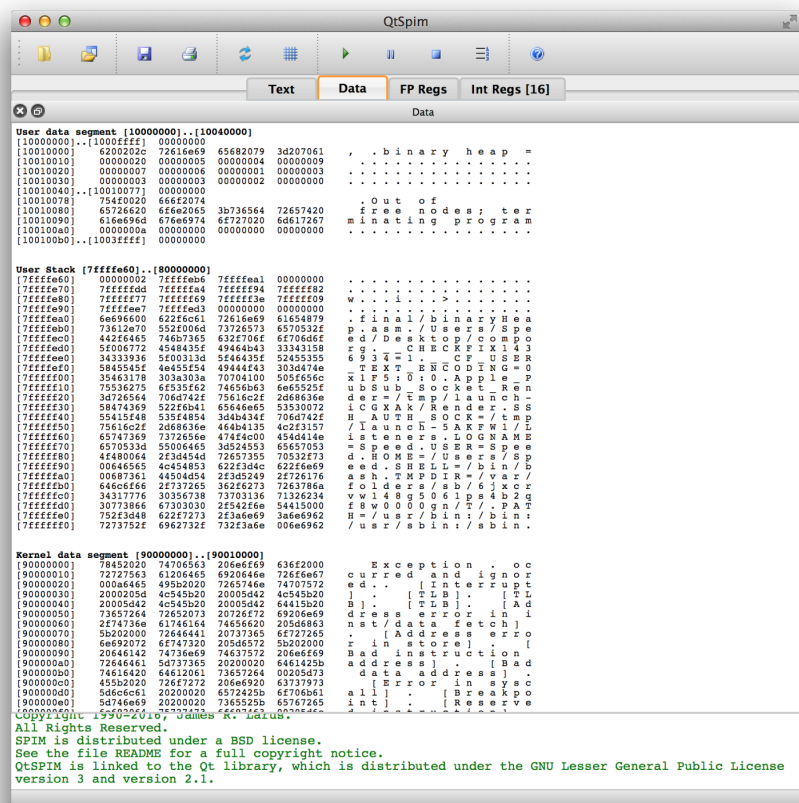


For my final project I decided to implement a binary heap in mips assembly. While being able to design any data structure in mips, I was most interested in the binary heap due to the lack of pointers used in this data structure. Throughout the term the projects we had to do in mips relied heavily on pointers, and I was interested in how a data type that used no pointers could be constructed. First, I had to decide what way I was going to store the data. Like a priority queue, I made the heap array-based, where the smallest elements are at the top of the heap and the biggest are at the bottom. This meant that for each index in the array i , the children of index i could be found at indexes $2*i+1$ and $2*i+2$. Similarly, at any index i , the parent of that index could be found at index $i/2$ if you rounded the remainder down.

There are two main ways to implement a minimum binary heap; recursively and iteratively. Because recursion is rather difficult in mips, I decided on the iterative method. Once I decided on an array based implementation with iterative subroutines, it was time to decide what the subroutines I would need to make this data structure would be. I needed a remove method that takes the minimum element from the heap, which also happens to be the first element, and returns the value. Similar to popping from the top of a stack, the minimum heap is only able to remove the smallest item from the top. I also need an insert method for adding new elements to the heap in the proper place. For both of these methods I had to devise a way to balance and restructure the heap after each insert and remove. To accomplish this, I made a `percUp` and `percDown` subroutine for each method respectively. Given a numbers index, `percUp` works by checking the indexes parent to see if the parent is larger or smaller than the child. If it is larger, then the two elements are swapped and the process is repeated. `PercUp` continues until the

parent of the number is smaller than it. PercDown does the opposite, starting from the index of a parent node, and checking to make sure the two children are bigger than it. If the minimum child of the parent index is smaller than the parent, the two are swapped. In order to make percDown, I decided to write a subroutine minChild. Given a parent index, minChild would look at the two children and return the smaller of the two. This was very useful for percDown as I didn't have to worry about finding the indexes of child nodes in the method. Once I had written the percDown method, I was ready to write the buildHeap method. Given an unsorted array, buildHeap would take the elements and insert them into a space allocated for the heap in the proper location by calling percDown after each insertion. Now that I had made a heap that could insert and remove, I added a find method, which given a number n, would check to see if the number was in the heap and return 1 if it was. While not necessary, It is always useful to have a find method in a data structure. Finally, I wrote a print method so the user could get an easy to look at view of the completed heap.

Data Before heaping:

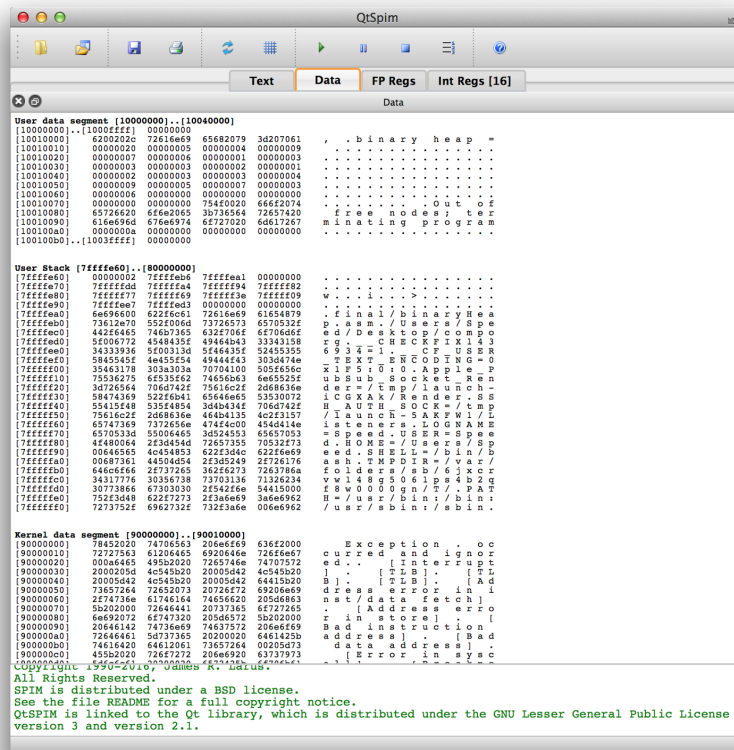


```
QtSpim
Text  Data  FP Regs  Int Regs [16]
Data
User data segment [10000000]..[10040000]
[10000000]..[1000ffff] 00000000
[10010000] 62002020 7261e6e9 65682079 3d207061 , .binary heap =
[10010010] 00000020 00000005 00000004 00000009 , . . . . .
[10010020] 00000007 00000006 00000001 00000003 , . . . . .
[10010030] 00000003 00000003 00000002 00000000 , . . . . .
[10010040]..[1001007f] 00000000
[10010078] 754f0020 666f2074 3b736564 72657420 , .Out of
[10010080] 65726620 656e2065 3b736564 72657420 , free nodes; ter
[10010090] 616e696d 676e6974 6f727020 6d617267 , minating program
[100100a0] 0000000a 00000000 00000000 00000000 , . . . . .
[100100b0]..[1003ffff] 00000000

User Stack [7ffffe60]..[80000000]
[7ffffe60] 00000002 7ffffeb6 7ffffea1 00000000 , . . . . .
[7ffffe70] 7ffffe6d 7ffffe94 7ffffe94 7ffffe92 , . . . . .
[7ffffe80] 7ffffe77 7ffffe69 7ffffe3e 7ffffe09 , . . . . .
[7ffffe90] 7ffffe67 7ffffe63 00000000 00000000 , . . . . .
[7ffffea0] 6e696e00 622f6e61 7561e6e9 615e4879 , . . . . .
[7ffffeb0] 73612e70 552e006d 73726573 6570532f , p.asm./Users/Sp
[7ffffec0] 442e64e5 746b7365 632f706f 6f706d6f , ed/Docker/comp
[7ffffed0] 5f006772 4548435f 49464b43 33343158 , rg. CHRECKFIX143
[7ffffee0] 34333936 5d00313d 5f46435f 52455355 , 693X=1. CFWERR
[7ffffef0] 58455455 4e455564 49464f43 3036474e , TEXTENCODING=0
[7fffff00] 35463178 303a303a 70704100 505f656c , XLF5:0:Apple_P
[7fffff10] 75536275 65535f62 74656b63 6e65525f , ubSub SocketRen
[7fffff20] 3d726564 706f742f 75616c2f 2d68636e , der/tmp/lanchn-
[7fffff30] 58474369 522f6b41 65464e65 53530072 , iCGXAK/Render.ss
[7fffff40] 59415f48 335f4854 3d4b434f 706d7422 , HAUTH:OCK-/tmp
[7fffff50] 75616c2f 2d68636e 464b4135 4c2f3157 , /TauncH-5AKFW1/L
[7fffff60] 65747369 7372656e 47424c00 454d414e , isteners.LOGNAME
[7fffff70] 65705338 55006465 3d524553 65657053 , =Speed. USERSpee
[7fffff80] 4f480064 2f3d454d 72657355 70532f73 , d.HOME=/Users/Sp
[7fffff90] 00646565 4c454853 622f3d6c 622f6e69 , eed.SHELL=/bin/b
[7fffffa0] 00687361 44504d54 2f3d5249 2f726176 , ash.TMPDIR=/var/
[7fffffb0] 646c6f66 2f737265 362f6273 7263786a , folders/ab/xcrcr
[7fffffc0] 34317776 303b6738 73703136 71302034 , vli8g561p4b2g
[7fffffd0] 30773866 67303030 2f542f6e 54415000 , f8w0000gn/T/.PAT
[7fffffe0] 752f3d48 622f7273 2f3a6e69 3a6e6962 , H/user/bin/bin:
[7ffffff0] 7273752f 6962732f 732f3a6e 00e6e962 , /usr/sbin/bin.

Kernel data segment [90000000]..[90010000]
[90000000] 78452020 74706563 206e6f69 636f2000 , Exception . oc
[90000010] 72172763 61206465 6920646e 726e6e67 , curred and signr
[90000020] 000a6465 495b2020 7265746e 74707572 , ed. [Interrupt
[90000030] 20002054 4c545b20 20005d42 4c540b20 , ]. [TLB] . [TL
[90000040] 20005d42 4c545b20 20005d42 64415b20 , B]. [TLB] . [Ad
[90000050] 73657264 72652073 20726f72 69206e69 , dress error in l
[90000060] 2f474736 61746164 74656620 205d6863 , nat/data fecb]
[90000070] 5b202000 72646441 20737365 6f727265 , . [Address erro
[90000080] 6e692072 6f747320 205d6862 5b202000 , i store] (
[90000090] 20546142 74736e69 74637572 206e6f69 , Bad instruction
[900000a0] 72646461 5d737365 20200020 6461425b , address. [Bad
[900000b0] 74616420 64612061 73657264 00205d73 , data address]
[900000c0] 455b2020 726f7272 206e6920 63737973 , [Error in sysc
[900000d0] 5d6c6e61 20200020 6572425b 6f706b61 , all] . [Breakpo
[900000e0] 5d746e69 20200020 73657265 65767265 , int] . [Reserv
[900000f0] 726f7272 6962732f 732f3a6e 00e6e962 , /usr/sbin/bin.

Copyright 1990-2010, Qantas Research, Inc.
All Rights Reserved.
SPM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License
version 3 and version 2.1.
```



```
[10000000]..[10000000]
[10000000] 00000000 00000000 00000000 00000000  . . . . .
[10000000] 62000020 7616e69 65682079 3d207061  . . . . .
[10000000] 00000020 00000005 00000004 00000009  . . . . .
[10000000] 00000007 00000006 00000001 00000003  . . . . .
[10000000] 00000003 00000003 00000002 00000001  . . . . .
[10000000] 00000002 00000003 00000003 00000004  . . . . .
[10000000] 00000009 00000005 00000007 00000003  . . . . .
[10000000] 00000006 00000000 00000000 00000000  . . . . .
[10000000] 00000000 00000000 754f0320 64622074  . . . . .
[10000000] 65726620 616e2065 3b736564 72657420  f r e e  m e m o r y  t e r
[10000000] 616e096d 676e0974 6f727320 6d617267  m i n i m u m  p r o g r a m
[10000000] 0000000a 00000000 00000000 00000000  . . . . .
[10000000]..[1003ffff] 00000000

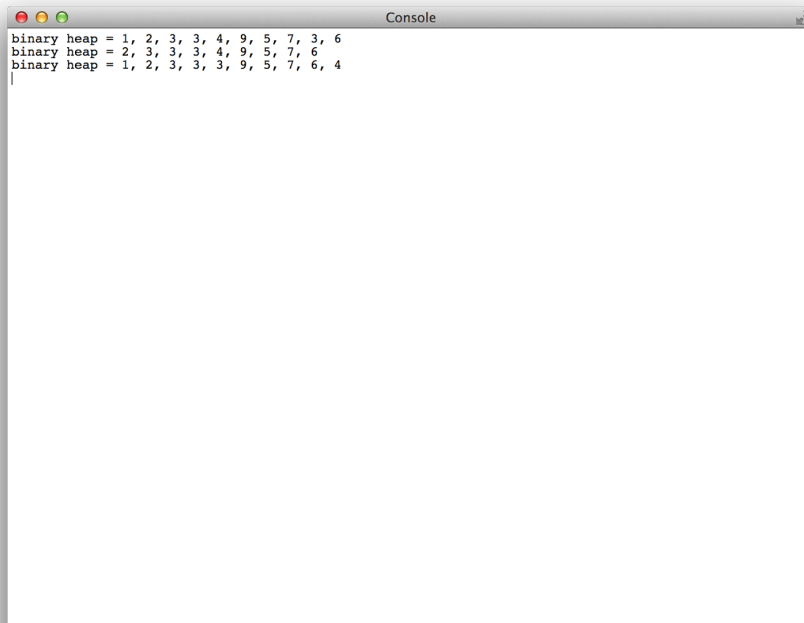
User Stack [7ffffe00]..[80000000]
[7ffffe00] 00000002 7ffffe06 7ffffe0a 00000000  . . . . .
[7ffffe00] 7ffffe06 7ffffe04 7ffffe04 7ffffe02  . . . . .
[7ffffe00] 7ffffe07 7ffffe09 7ffffe0e 7ffffe09  w . . i . . > . . . . .
[7ffffe00] 7ffffe07 7ffffe03 00000000 00000000  . . . . .
[7ffffe00] 6e696000 622f6c61 72616e69 61654879  . . . . .
[7ffffe00] 73612e70 552f006d 73726573 6570532f  p . a s s . / u s e r / s p e
[7ffffe00] 44226465 74607365 6327706f 67096d6f  e d / d e s t o p / c o m p o
[7ffffe00] 5f006772 4548435f 49464b43 33343158  r g . . . C R E C K F I X 1 4 3
[7ffffe00] 34333936 5000313d 5f46435f 52455355  6 9 3 4 - 1  C P U S E R
[7ffffe00] 5845545f 4e455454 49444f43 3036474e  T E X T  E N C O D I N G = 0
[7ffffe00] 35463178 303a303a 70704100 5050506e  x l f s i o . A p p l e P
[7ffffe00] 75536275 65535662 74656b63 6e65525f  u b s u b s o c k e t . s e n
[7ffffe00] 3d726564 706d742f 75616c2f 2d68636e  d e r = / t m p / l a u n c h -
[7ffffe00] 58474369 522f6b41 65464e05 53530072  . C O X A K / R e n d e r . S S
[7ffffe00] 55415e40 535f4854 3d4b434f 706d742f  H A U T H . S O C K = / t m p
[7ffffe00] 75616c2f 2d68636e 464b4135 4c253157  / l a u n c h . S A R F W / L
[7ffffe00] 65747369 7372656e 474f4c00 4546414e  i s t e n e r s . L O G N A M E
[7ffffe00] 6570533d 53004645 3d524553 65657053  . s p e e d . U S E R = s p e e
[7ffffe00] 4f480964 2f3d464d 72657355 70524773  e . H O M E / u s e r s / s p
[7ffffe00] 00646565 4c454853 622f3d4c 622f6e69  e e d . S H E L L = / b i n / b
[7ffffe00] 00687361 4d504654 2f3d5248 2f726176  . s h . T M P D I R / v e r /
[7ffffe00] 646c6f66 2f737265 3d2f6273 7263786a  f o l d e r s / s b / 6 3 x c r
[7ffffe00] 3d277776 303d6738 73703136 71326234  v w l 4 9 s 0 6 1 p e 4 2 q
[7ffffe00] 30773866 67303030 2f542f6e 54415000  F 8 W 0 0 0 0 g n / F . p A T
[7ffffe00] 75213d48 622f7273 2f3d4e69 3d6e6962  H = / u s r / b i n / : b i n :
[7ffffe00] 7273726f 6867736f 73723d4e 006e6962  / u s r / s b i n / s b i n .

Kernel data segment [90000000]..[90010000]
[90000000] 7852020 74705653 206e4e69 636f2000  . . . . .
[90000000] 72727563 61206465 6920646e 726f6e67  c u r r e d a n d i g n o r
[90000000] 000a4465 495b2020 7265746e 74707572  e d . . [ i n t e r r u p t
[90000000] 2000206d 4c455b20 20005d42 4c455b20  . . . [ T L B ] . [ T L
[90000000] 20005d42 4c455b20 20005d42 4c455b20  . . . [ T L B ] . [ A d
[90000000] 73657264 72652073 20726272 69206e69  d r e s s e r o r [ i
[90000000] 2f74736e 61746164 74656620 205d6863  n s t / d a t a f e t c h ]
[90000000] 3b202000 72646441 20737365 6f727265  . . . [ A d d r e s s e r r o
[90000000] 6e692072 6e747320 205d6572 5b202000  . . . r i n s t o r e ] . [
[90000000] 20646142 74736e69 74637572 206e6f69  . . . B a d i n s t r u c t i o n
[90000000] 72646461 5d777365 20200020 6461425b  . . . a d d r e s s ] . [ B a d
[90000000] 74616420 64612061 73657264 00205d73  . . . d a t a a d d r e s s .
[90000000] 455b2020 72677272 206e6920 63737573  . . . [ i n t e r r u p t a n d e r r o
[90000000] 00000000 00000000 00000000 00000000  . . . . .

Copyright 1995-2000 by Apple Computer, Inc.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License
version 3 and version 2.1.
```

Data After heaping

Console after buildHeap, RemoveMin, and Insert 1



```
binary heap = 1, 2, 3, 3, 3, 4, 9, 5, 7, 3, 6
binary heap = 2, 3, 3, 3, 3, 4, 9, 5, 7, 6
binary heap = 1, 2, 3, 3, 3, 9, 5, 7, 6, 4
```