lab4 test&review

陈俊卉 学号：2020212256 班级：2020219111

Task 1: Threads and the ThreadManager

1、代码解释

① Init

```
      ----------  ThreadManager . Init  ----------

      method Init ()
        --
        -- This method is called once at kernel startup time to initialize
        -- the one and only "ThreadManager" object.
        --
        var
          i: int



          print ("Initializing Thread Manager...\n")

```

```
            -- initialize the array of threads
            self.threadTable = new array of Thread {10 of new Thread}

            -- initialize each thread
            self.threadTable[0].Init("0")
            self.threadTable[1].Init("1")
            self.threadTable[2].Init("2")
            self.threadTable[3].Init("3")
            self.threadTable[4].Init("4")
            self.threadTable[5].Init("5")
            self.threadTable[6].Init("6")
            self.threadTable[7].Init("7")
            self.threadTable[8].Init("8")
            self.threadTable[9].Init("9")


            -- initialize freelist
            self.freeList = new List [Thread]

            for i = 0 to 9
              self.threadTable[i].status = UNUSED
              self.freeList.AddToEnd (&threadTable[i])
            endFor


            -- initialize mutex and condition
            self.threadManagerLock = new Mutex
            self.aThreadBecameFree = new Condition

            self.threadManagerLock.Init()
            self.aThreadBecameFree.Init()

          endMethod
```

解释：简单的初始化。

② GetANewThread

```
        ----------  ThreadManager . GetANewThread  ----------

        method GetANewThread () returns ptr to Thread
          --
          -- This method returns a new Thread; it will wait
          -- until one is available.
          --
          var th: ptr to Thread


           -- lock
           self.threadManagerLock.Lock()

           while freeList.IsEmpty() == true
             aThreadBecameFree.Wait(&self.threadManagerLock)
           endWhile

           -- remove and return_ a thread from freelist
```

```
            th = freeList.Remove()


            th.status = JUST_CREATED


            -- unlock
            self.threadManagerLock.Unlock()

            return th
         endMethod
```

解释：

- 因为GetANewThread是入口方法，所以需要在一开始获得锁，在最后释放锁。
- GetANewThread需要在freeList中删除并返回一个线程，所以如果freeList为空，我们需要等待线程条件满足再继续执行。而由于condition是mesa-style的，条件变量signal(broadcast)后并不是立即执行，而需要再次进行条件判断，所以我们需要使用while，而不是if.
- 将线程从freeList取出后，我们需要将其状态设为JUST_CREATED.
- 最后释放锁。

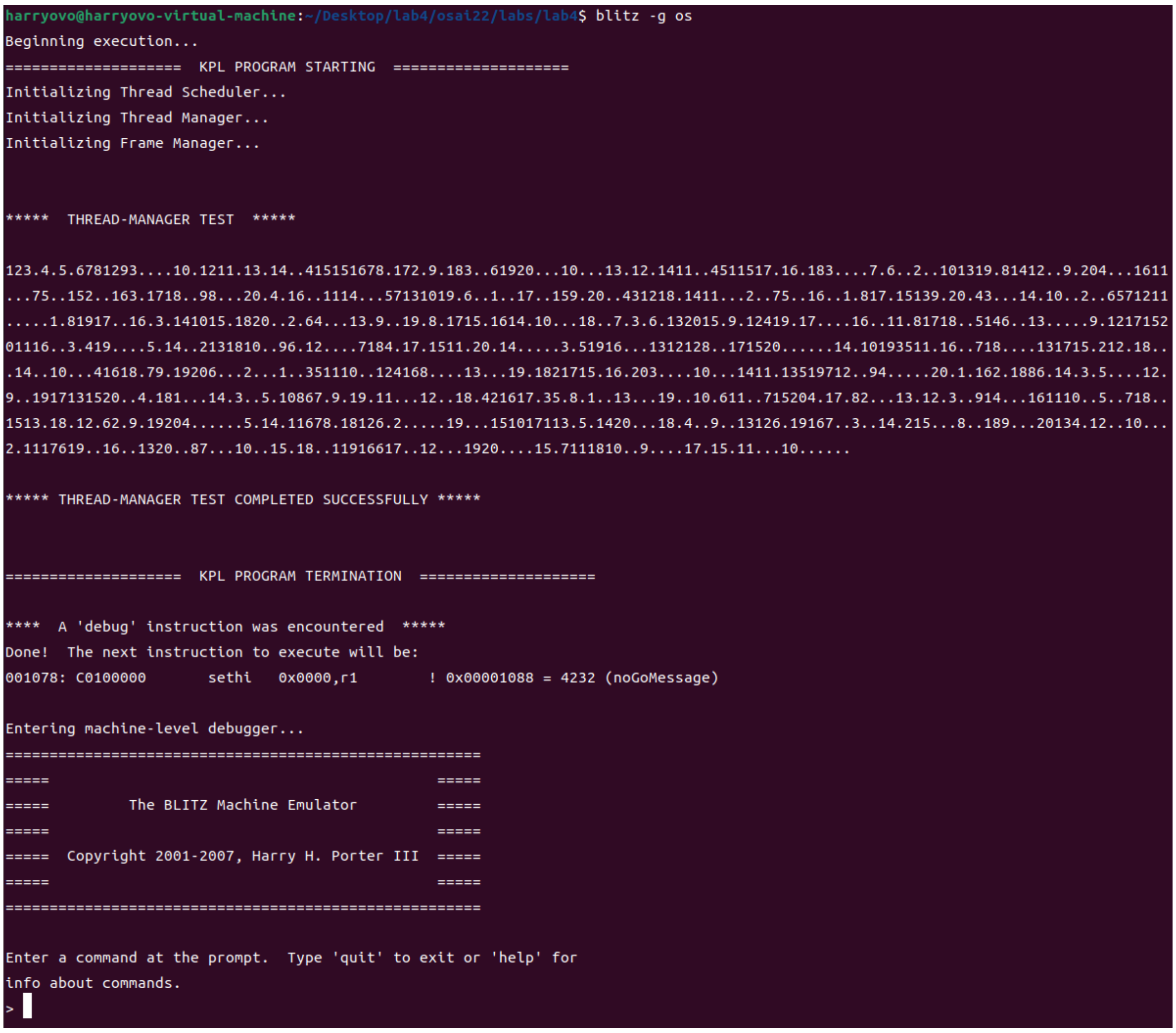
③ FreeThread

```
      method FreeThread (th: ptr to Thread)
        --
        -- This method is passed a ptr to a Thread;  It moves it
        -- to the FREE list.
        --

           -- lock
           self.threadManagerLock.Lock()



           th.status = UNUSED

           freeList.AddToEnd (th)
           aThreadBecameFree.Broadcast(&self.threadManagerLock)


           -- unlock
           self.threadManagerLock.Unlock()


        endMethod

    endBehavior
```

解释：

- 因为FreeThread是入口方法，所以需要在一开始获得锁，在最后释放锁。
- 需要将该线程的状态置为UNUSED.
- 然后将其放入freeList中。
- 依据pdf，需要向所有等待该条件的人发出信号，故这里使用broadcast.
- 最后释放锁。

2、输出



```
harryovo@harryovo-virtual-machine:~/Desktop/lab4/osai22/labs/lab4$ blitz -g os
Beginning execution...
====================  KPL PROGRAM STARTING  ====================
Initializing Thread Scheduler...
Initializing Thread Manager...
Initializing Frame Manager...


*****  THREAD-MANAGER TEST  *****

123.4.5.6781293....10.1211.13.14..415151678.172.9.183..61920...10...13.12.1411..4511517.16.183....7.6..2..101319.81412..9.204...1611
...75..152..163.1718..98...20.4.16..1114...57131019.6..1..17..159.20..431218.1411...2..75..16..1.817.15139.20.43...14.10..2..6571211
.....1.81917..16.3.141015.1820..2.64...13.9..19.8.1715.1614.10...18..7.3.6.132015.9.12419.17....16..11.81718..5146..13.....9.1217152
01116..3.419....5.14..2131810..96.12....7184.17.1511.20.14.....3.51916...1312128..171520......14.10193511.16..718....131715.212.18..
.14..10...41618.79.19206...2...1..351110..124168....13...19.1821715.16.203....10...1411.13519712..94.....20.1.162.1886.14.3.5....12.
9..1917131520..4.181...14.3..5.10867.9.19.11...12..18.421617.35.8.1..13...19..10.611..715204.17.82...13.12.3..914...161110..5..718..
1513.18.12.62.9.19204......5.14.11678.18126.2.....19...151017113.5.1420...18.4..9..13126.19167..3..14.215...8..189...20134.12..10...
2.1117619..16..1320..87...10..15.18..11916617..12...1920....15.7111810..9....17.15.11...10......

***** THREAD-MANAGER TEST COMPLETED SUCCESSFULLY *****


====================  KPL PROGRAM TERMINATION  ====================

****  A 'debug' instruction was encountered  *****
Done!  The next instruction to execute will be:
001078: C0100000       sethi   0x0000,r1        ! 0x00001088 = 4232 (noGoMessage)

Entering machine-level debugger...
======================================================
=====                                            =====
=====         The BLITZ Machine Emulator         =====
=====                                            =====
====  Copyright 2001-2007, Harry H. Porter III  =====
=====                                            =====
======================================================

Enter a command at the prompt.  Type 'quit' to exit or 'help' for
info about commands.
> 
```

Task 2: Processes and the ProcessManager

1、代码解释

① Init

```
      ----------  ProcessManager . Init  ----------

      method Init ()
        --
        -- This method is called once at kernel startup time to initialize
        -- the one and only "processManager" object.
        --
        var
          i: int

            self.processTable = new array of ProcessControlBlock {10 of new ProcessControlBlock}


            for i = 0 to 9
              self.processTable[i].Init()
            endFor


            self.processManagerLock = new Mutex
            self.processManagerLock.Init()

            self.aProcessBecameFree = new Condition
            self.aProcessBecameFree.Init()

            self.aProcessDied = new Condition
            self.aProcessDied.Init()

            self.freeList = new List [ProcessControlBlock]


            for i = 0 to 9
              self.freeList.AddToEnd (&processTable[i])
            endFor

            self.nextPid = 0


        -- NOT IMPLEMENTED
        endMethod
```

解释：初始化，pid将会在GetANewProcess初始化。

② GetANewProcess

```
      ----------  ProcessManager . GetANewProcess  ----------

      method GetANewProcess () returns ptr to ProcessControlBlock
        --
        -- This method returns a new ProcessControlBlock; it will wait
        -- until one is available.
        --
          -- NOT IMPLEMENTED
          var pr :ptr to ProcessControlBlock

          -- lock
          self.processManagerLock.Lock()

          while freeList.IsEmpty() == true
            aProcessBecameFree.Wait(&self.processManagerLock)
          endWhile

          -- remove and return_ a process from freelist
          pr = freeList.Remove()

          self.nextPid = self.nextPid + 1
          pr.pid = self.nextPid

          pr.status = ACTIVE

          -- unlock
          self.processManagerLock.Unlock()


          return pr
        endMethod
```

解释:

- GetANewProcess需要对pid进行分配。而分配的数字是ProcessManager所保存的nextPid.
- 同样地，GetANewProcess是入口方法，所以需要在一开始获得锁，在最后释放锁。
- GetANewProcess的结构与GetANewThread基本一致：freeList为空时需要等待、满足条件后从freeList拿出相应process，更新nextPid后为该process设置pid，并将其的status设为ACTIVE.
- 最后释放锁。

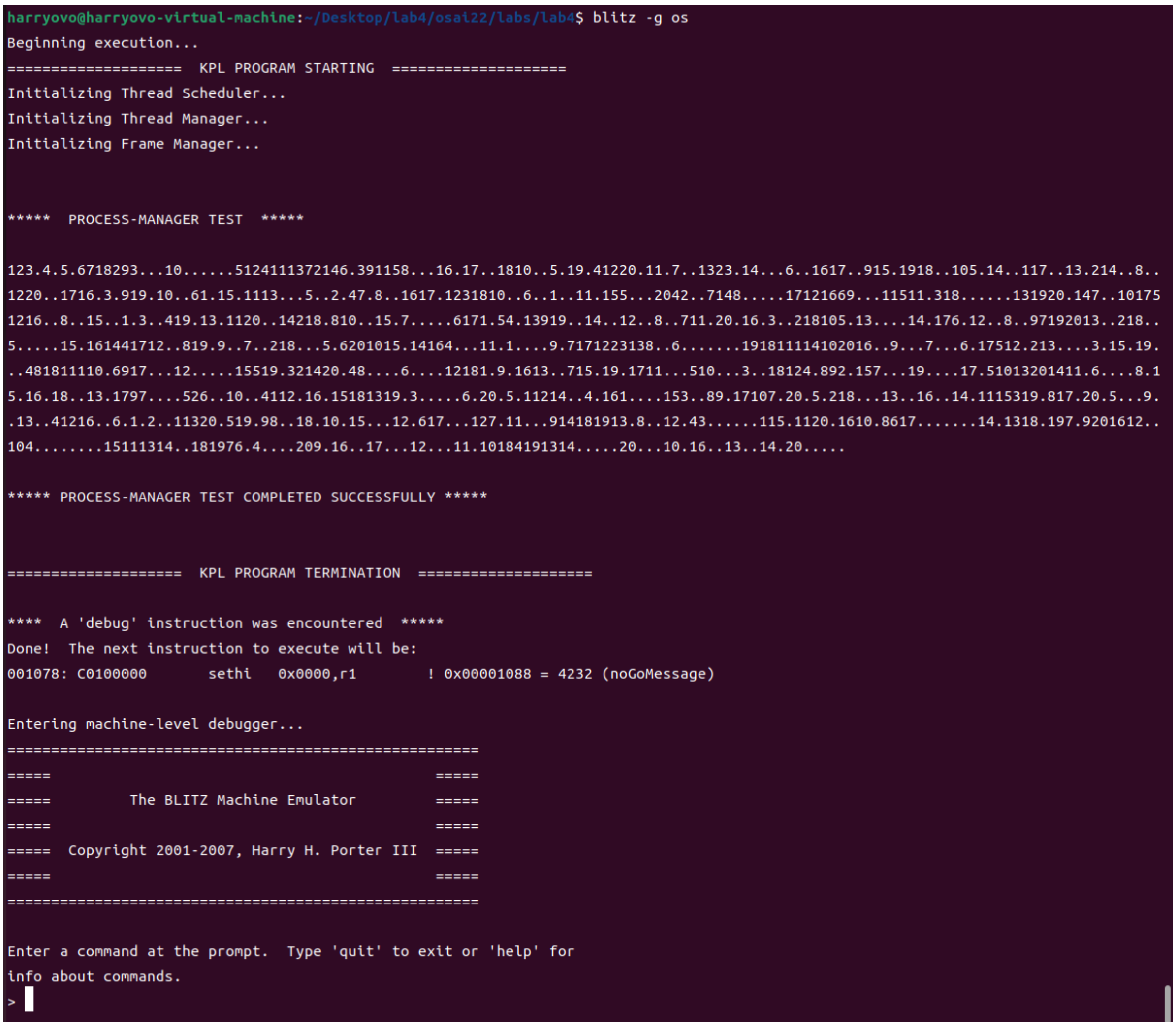
③ FreeProcess

```
      ----------  ProcessManager . FreeProcess  ----------

       method FreeProcess (p: ptr to ProcessControlBlock)
         --
         -- This method is passed a ptr to a Process;  It moves it
         -- to the FREE list.
         --
           -- NOT IMPLEMENTED


           -- lock
            self.processManagerLock.Lock()


            p.status = FREE
            p.pid = -1

            freeList.AddToEnd (p)
            aProcessBecameFree.Broadcast(&self.processManagerLock)



            -- unlock
            self.processManagerLock.Unlock()


         endMethod
```

解释:

- FreeProcess是入口方法，所以需要在一开始获得锁，在最后释放锁。
- 我们需要将该process的状态设为FREE，并将pid清空，加入freeList.
- 依据pdf，需要向所有等待该条件的人发出信号，故这里使用broadcast.
- 最后释放锁。

2、输出：



```
harryovo@harryovo-virtual-machine:~/Desktop/lab4/osai22/labs/lab4$ blitz -g os
Beginning execution...
====================  KPL PROGRAM STARTING  ====================
Initializing Thread Scheduler...
Initializing Thread Manager...
Initializing Frame Manager...


*****  PROCESS-MANAGER TEST  *****

123.4.5.6718293...10......5124111372146.391158...16.17..1810..5.19.41220.11.7..1323.14...6..1617..915.1918..105.14..117..13.214..8..
1220..1716.3.919.10..61.15.1113...5..2.47.8..1617.1231810..6..1..11.155...2042..7148.....17121669...11511.318......131920.147..10175
1216..8..15..1.3..419.13.1120..14218.810..15.7.....6171.54.13919..14..12..8..711.20.16.3..218105.13....14.176.12..8..97192013..218..
5.....15.161441712..819.9..7..218...5.6201015.14164...11.1....9.7171223138..6.......191811114102016..9...7...6.17512.213....3.15.19.
..481811110.6917...12.....15519.321420.48....6....12181.9.1613..715.19.1711...510...3..18124.892.157...19....17.51013201411.6....8.1
5.16.18..13.1797....526..10..4112.16.15181319.3.....6.20.5.11214..4.161....153..89.17107.20.5.218...13..16..14.1115319.817.20.5...9.
.13..41216..6.1.2..11320.519.98..18.10.15...12.617...127.11...914181913.8..12.43......115.1120.1610.8617.......14.1318.197.9201612..
104........15111314..181976.4....209.16..17...12...11.10184191314.....20...10.16..13..14.20.....

***** PROCESS-MANAGER TEST COMPLETED SUCCESSFULLY *****


====================  KPL PROGRAM TERMINATION  ====================

****  A 'debug' instruction was encountered  *****
Done!  The next instruction to execute will be:
001078: C0100000       sethi   0x0000,r1        ! 0x00001088 = 4232 (noGoMessage)

Entering machine-level debugger...
======================================================
=====                                            =====
=====         The BLITZ Machine Emulator         =====
=====                                            =====
=====  Copyright 2001-2007, Harry H. Porter III  =====
=====                                            =====
======================================================

Enter a command at the prompt.  Type 'quit' to exit or 'help' for
info about commands.
> 
```

Task 3: The Frame Manager

1、代码解释

① GetNewFrames

```
      ----------  FrameManager . GetNewFrames  ----------

      method GetNewFrames (aPageTable: ptr to AddrSpace, numFramesNeeded: int)
          -- NOT IMPLEMENTED
          var
            free_frame_index: int
            free_frame_addr: int
            i: int

          frameManagerLock.Lock()

          wait_count = wait_count + 1
          if wait_count > 1
            wait_.Wait(&frameManagerLock)
          endIf

          while numberFreeFrames < numFramesNeeded
            newFramesAvailable.Wait(&frameManagerLock)
          endWhile

          -- now available

          for i = 0 to numFramesNeeded - 1
            free_frame_index = framesInUse.FindZeroAndSet()
            free_frame_addr = PHYSICAL_ADDRESS_OF_FIRST_PAGE_FRAME +
(free_frame_index * PAGE_SIZE)
            aPageTable.SetFrameAddr (i, free_frame_addr)
          endFor

          numberFreeFrames = numberFreeFrames - numFramesNeeded

          aPageTable.numberOfPages = numFramesNeeded


          -- add
          wait_count = wait_count - 1
          wait_.Signal(&frameManagerLock)


          frameManagerLock.Unlock()


        endMethod
```

解释：

- 首先上锁。
- 根据pdf所说：
  - You'll need to do a Broadcast, because a Signal will only wake up one thread. The thread that gets
    awakened may not have enough free frames to complete, but other waiting threads may be able to
    proceed. A broadcast should be adequate, but perhaps after carefully studying the Game Parlor problem,
    you will find a more elegant approach which wakes up only a single thread.
  - 为了防止饥饿，笔者迁移了在Game Parlor problem中解决饥饿的**"优雅的方法"**：使用额外的一个条件变量wait_，**以保证每次进入while判断的进程只有一个，不会发生争抢的情况。** wait_count用来累计当前等待的进程数。只要这个数大于1，后来的进程都要进入条件变量wait_的等待队列中，直到前面已经进入while判断的进程满足条件，被成功分配帧后，条件变量wait_才会被signal。
- 当可用帧大于所需帧，即成功分配后，需要在framesInUse这个BitMap中下标由小到大找到可用帧并计算出相对应的地址，存储已分配帧的地址。
- 随后numberFreeFrames减去刚刚分配的帧数，更新aPageTable.numberOfPages的值。
- 该进程的分配已经结束，等待分配的数量减1，同时对wait_进行signal，下一个进程进入分配帧的条件判断。
- 最后释放锁。

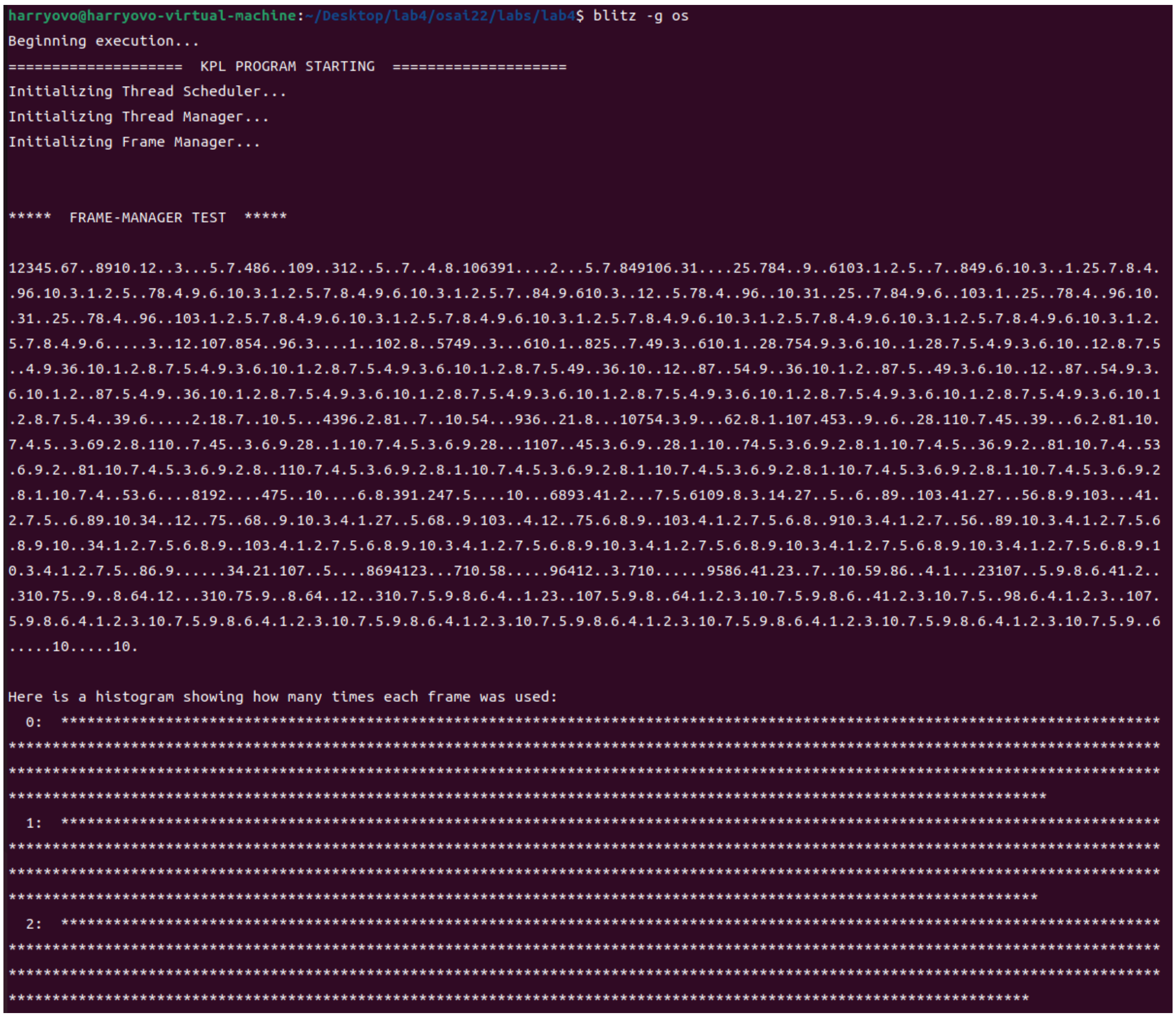
② ReturnAllFrames

```
      ----------  FrameManager . ReturnAllFrames  ----------

      method ReturnAllFrames (aPageTable: ptr to AddrSpace)
          -- NOT IMPLEMENTED
          var
            numFramesReturned: int
            frameAddr: int
            bitNumber: int
            i: int


          frameManagerLock.Lock()

          numFramesReturned = aPageTable.numberOfPages

          for i = 0 to numFramesReturned - 1
            frameAddr = aPageTable.ExtractFrameAddr(i)
            bitNumber = (frameAddr -  PHYSICAL_ADDRESS_OF_FIRST_PAGE_FRAME)
/  PAGE_SIZE
            framesInUse.ClearBit(bitNumber)
          endFor

          -- ??? whether needing to set 0?
          -- A:need to but not neccessary:it may be covered by next
need_num.
          --aPageTable.numberOfPages = 0

          numberFreeFrames = numberFreeFrames + numFramesReturned

          newFramesAvailable.Signal(&frameManagerLock)


          frameManagerLock.Unlock()

        endMethod
```

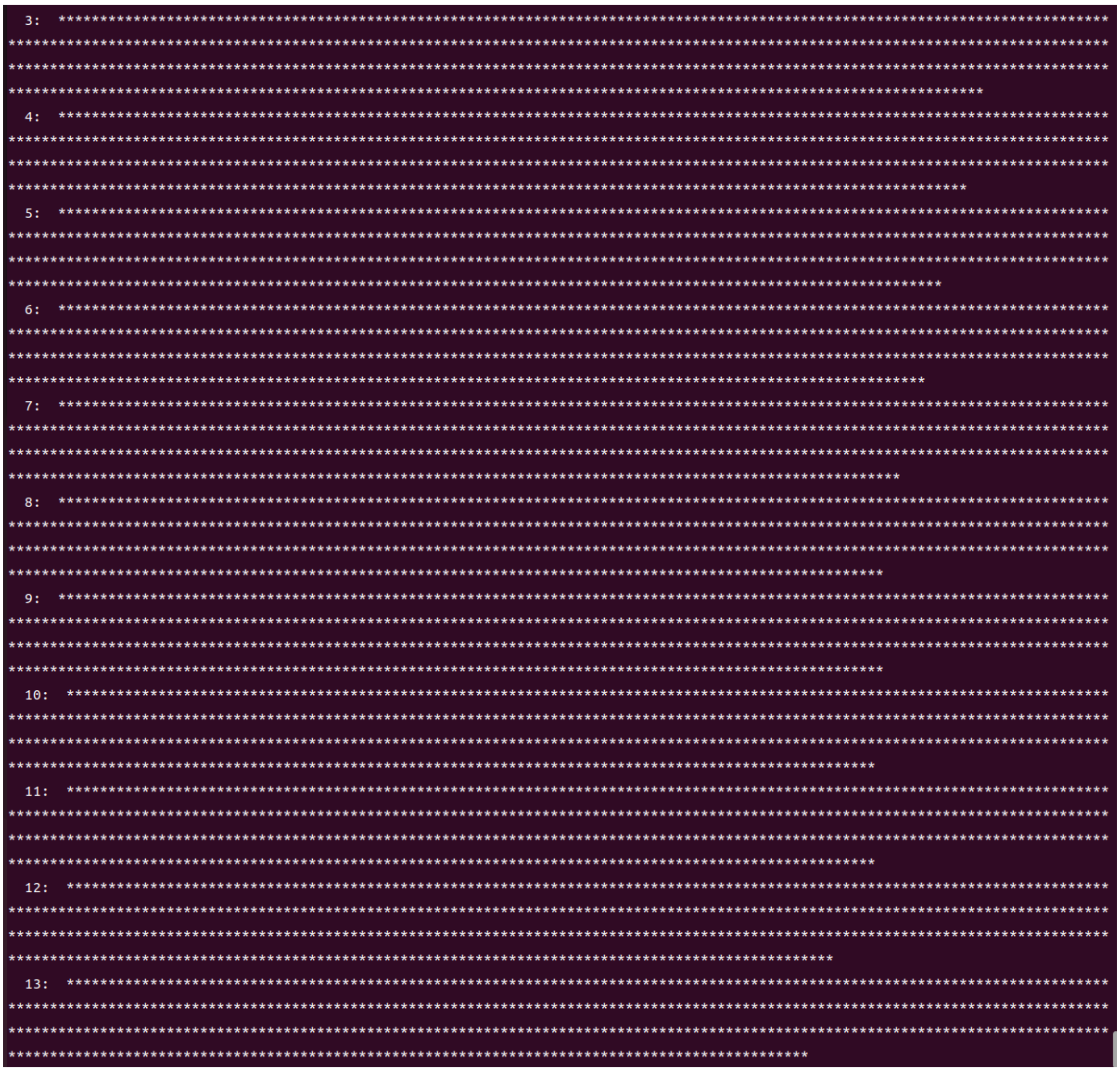
解释：

- 首先上锁。
- 首先获取将要释放的帧数。
- 随后一个接一个地获取要释放的帧的地址，并转化为对应的bitmap上的bitNumber，最后clear.
- 理论上，此时aPageTable.numberOfPages需要置零。但考虑到之后重新申请帧时会直接覆盖值，所以不必要。
- 更新numberFreeFrames，并对条件变量newFramesAvailable进行signal.（但不代表立刻执行，还需要进行再次的条件判断——即可用帧是否大于所需帧）
- 最后释放锁。

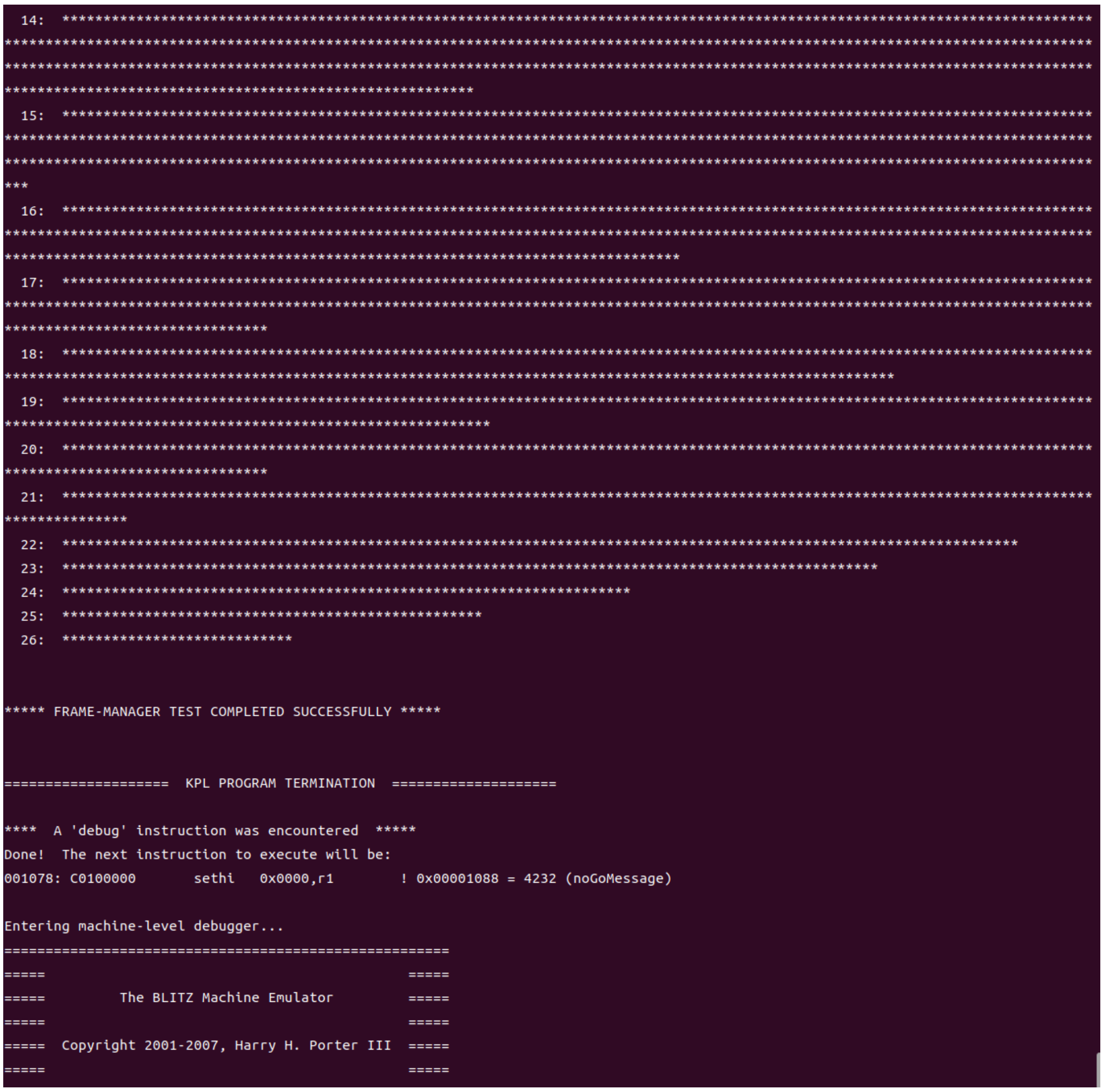
2、输出：



```
harryovo@harryovo-virtual-machine:~/Desktop/lab4/osai22/labs/lab4$ blitz -g os
Beginning execution...
====================  KPL PROGRAM STARTING  ====================
Initializing Thread Scheduler...
Initializing Thread Manager...
Initializing Frame Manager...


*****  FRAME-MANAGER TEST  *****

12345.67..8910.12..3...5.7.486..109..312..5..7..4.8.106391....2...5.7.849106.31....25.784..9..6103.1.2.5..7..849.6.10.3..1.25.7.8.4.
.96.10.3.1.2.5..78.4.9.6.10.3.1.2.5.7.8.4.9.6.10.3.1.2.5.7..84.9.610.3..12..5.78.4..96..10.31..25..7.84.9.6..103.1..25..78.4..96.10.
.31..25..78.4..96..103.1.2.5.7.8.4.9.6.10.3.1.2.5.7.8.4.9.6.10.3.1.2.5.7.8.4.9.6.10.3.1.2.5.7.8.4.9.6.10.3.1.2.5.7.8.4.9.6.10.3.1.2.
5.7.8.4.9.6.....3..12.107.854..96.3....1..102.8..5749..3...610.1..825..7.49.3..610.1..28.754.9.3.6.10..1.28.7.5.4.9.3.6.10..12.8.7.5
..4.9.36.10.1.2.8.7.5.4.9.3.6.10.1.2.8.7.5.4.9.3.6.10.1.2.8.7.5.49..36.10..12..87..54.9..36.10.1.2..87.5..49.3.6.10..12..87..54.9.3.
6.10.1.2..87.5.4.9..36.10.1.2.8.7.5.4.9.3.6.10.1.2.8.7.5.4.9.3.6.10.1.2.8.7.5.4.9.3.6.10.1.2.8.7.5.4.9.3.6.10.1.2.8.7.5.4.9.3.6.10.1
.2.8.7.5.4..39.6.....2.18.7..10.5...4396.2.81..7..10.54...936..21.8...10754.3.9...62.8.1.107.453..9..6..28.110.7.45..39...6.2.81.10.
7.4.5..3.69.2.8.110..7.45..3.6.9.28..1.10.7.4.5.3.6.9.28...1107..45.3.6.9..28.1.10..74.5.3.6.9.2.8.1.10.7.4.5..36.9.2..81.10.7.4..53
.6.9.2..81.10.7.4.5.3.6.9.2.8..110.7.4.5.3.6.9.2.8.1.10.7.4.5.3.6.9.2.8.1.10.7.4.5.3.6.9.2.8.1.10.7.4.5.3.6.9.2.8.1.10.7.4.5.3.6.9.2
.8.1.10.7.4..53.6....8192....475..10....6.8.391.247.5....10...6893.41.2...7.5.6109.8.3.14.27..5..6..89..103.41.27...56.8.9.103...41.
2.7.5..6.89.10.34..12..75..68..9.10.3.4.1.27..5.68..9.103..4.12..75.6.8.9..103.4.1.2.7.5.6.8..910.3.4.1.2.7..56..89.10.3.4.1.2.7.5.6
.8.9.10..34.1.2.7.5.6.8.9..103.4.1.2.7.5.6.8.9.10.3.4.1.2.7.5.6.8.9.10.3.4.1.2.7.5.6.8.9.10.3.4.1.2.7.5.6.8.9.10.3.4.1.2.7.5.6.8.9.1
0.3.4.1.2.7.5..86.9......34.21.107..5....8694123...710.58.....96412..3.710......9586.41.23..7..10.59.86..4.1...23107..5.9.8.6.41.2..
.310.75..9..8.64.12...310.75.9..8.64..12..310.7.5.9.8.6.4..1.23..107.5.9.8..64.1.2.3.10.7.5.9.8.6..41.2.3.10.7.5..98.6.4.1.2.3..107.
5.9.8.6.4.1.2.3.10.7.5.9.8.6.4.1.2.3.10.7.5.9.8.6.4.1.2.3.10.7.5.9.8.6.4.1.2.3.10.7.5.9.8.6.4.1.2.3.10.7.5.9.8.6.4.1.2.3.10.7.5.9..6
.....10.....10.

Here is a histogram showing how many times each frame was used:
   0: ******************************************************************************************************************************
************************************************************************************************************************************
************************************************************************************************************************************
***********************************************************************************************************************
   1: ******************************************************************************************************************************
************************************************************************************************************************************
************************************************************************************************************************************
**********************************************************************************************************************
   2: ******************************************************************************************************************************
************************************************************************************************************************************
************************************************************************************************************************************
*********************************************************************************************************************
```



```
  3:  ********************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
***********************************************************************************************************************
  4:  ********************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
*********************************************************************************************************************
  5:  ********************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
******************************************************************************************************************
  6:  ********************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
****************************************************************************************************************
  7:  ********************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
*************************************************************************************************************
  8:  ********************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
***********************************************************************************************************
  9:  ********************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
***********************************************************************************************************
  10:  *******************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
**********************************************************************************************************
  11:  *******************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
**********************************************************************************************************
  12:  *******************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
*****************************************************************************************************
  13:  *******************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************************************************
**************************************************************************************************
```



```
  14:  ***************************************************************************************************************************
**********************************************************************************************************************************
**********************************************************************************************************************************
********************************************************
  15:  ***************************************************************************************************************************
**********************************************************************************************************************************
**********************************************************************************************************************************
***
  16:  ***************************************************************************************************************************
**********************************************************************************************************************************
********************************************************************************
  17:  ***************************************************************************************************************************
**********************************************************************************************************************************
*******************************
  18:  ***************************************************************************************************************************
**********************************************************************************************************
  19:  ***************************************************************************************************************************
**********************************************************
  20:  ***************************************************************************************************************************
*******************************
  21:  ***************************************************************************************************************************
***************
  22:  ******************************************************************************************************************
  23:  *************************************************************************************************
  24:  ********************************************************************
  25:  **************************************************
  26:  ***************************


***** FRAME-MANAGER TEST COMPLETED SUCCESSFULLY *****


===================  KPL PROGRAM TERMINATION  ===================

****  A 'debug' instruction was encountered  *****
Done!  The next instruction to execute will be:
001078: C0100000       sethi   0x0000,r1        ! 0x00001088 = 4232 (noGoMessage)

Entering machine-level debugger...
=====================================================
=====                                           =====
=====         The BLITZ Machine Emulator        =====
=====                                           =====
=====  Copyright 2001-2007, Harry H. Porter III  =====
=====                                           =====
```

Task 4: Change Condition Variables to Hoare Semantics

1、分析

- MESA semantics 和 Hoare Semantics 的区别：
  - Hoare Semantics：有一个**入口等待队列**以便管程外面的进程等待。在管程内有条件变量，若进程等待并释放互斥权，则在该条件变量上等待；在管程内有**紧急等待队列**，等待的进程进入该队列中，优先级高于入口等待队列。**使用if进行条件的判断，当紧急等待队列被signal，则立即执行。**由于使用if，所以Hoare不可能有broadcast方法，否则将产生混乱，并且，在signal前，还需要将锁交给被signal的进程,而不是解锁，以确保signal之后不会发生竞争，只有唯一一个进程会响应并接受条件。
  - MESA semantics：只有一个队列，当队列signal时，通知队头进程，但此时还不一定满足条件，可能仍然需要进入等待队列。由于被signal时不一定满足条件，所以需要使用while进行条件判断（有可能发生竞争从而使得条件仍然不满足），确保满足条件才进行下一步操作。

事实上Hoare Semantics跟我们在task3采用的“优雅的方法”有点相似。

2、个别代码的修改

① mutex需要增加Give方法，在signal之前将锁交给刚刚唤醒的进程，确保signal之后不会发生竞争，只有唯一一个进程会响应并接受条件。

```
      -----------  Mutex . Give  -----------

    method Give (t: ptr to Thread)
          var
             oldIntStat: int
          oldIntStat = SetInterruptsTo (DISABLED)
          if heldBy != currentThread
             FatalError ("Attempt to give away the mutex by a thread not 
holding it")
          endIf
          heldBy = t
          oldIntStat = SetInterruptsTo (oldIntStat)
        endMethod
```

- 首先需要关中断，因为这是原语操作！
- 将当前的锁的归属交给t.·
- 开中断。

② condition需要在将进程置为ready态后，当前进程将锁给予该进程。

```
     ----------  Condition . Signal  ----------

      method Signal (mutex: ptr to HoareMutex)
          var
            oldIntStat: int
                 t: ptr to Thread
          oldIntStat = SetInterruptsTo (DISABLED)
      if !mutex. IsHeldByCurrentThread ()
         FatalError ("Attempt to wait on condition when mutex is not held")
          endIf
          t = waitingThreads.Remove ()
          if t
             t.status = READY

             -- !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
             mutex.Give (t)

             readyList.AddToEnd (t)
      endIf
        oldIntStat = SetInterruptsTo (oldIntStat)
      endMethod

  endBehavior
```

③ Condition 的 Wait

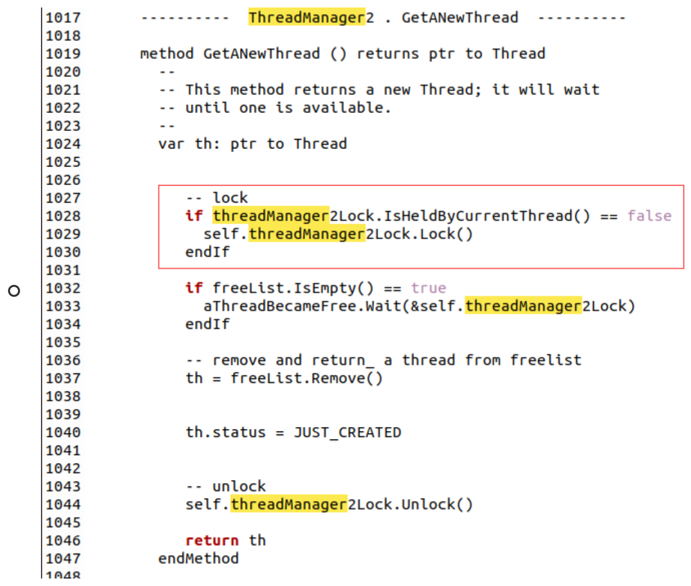
condition的Wait中，在唤醒并不需要重新上锁，因为signal后锁的heldBy被Give到刚刚被signal的进程，相当于保证了signal后下一个进程一定是这个刚刚唤醒的进程，而不是排在Lock队列中较前的进程（如果不give，且跟MESA一样只在唤醒后Lock，则当前进程只能排在Lock队列的后面，仍然可能被其他进程抢先占用资源）。前面的进程如果试图Lock，将会在Lock队列中排在刚刚唤醒的这个进程的后面。

```
      ----------  Condition . Wait  ----------

      method Wait (mutex: ptr to Mutex)
          var
            oldIntStat: int
          if ! mutex.IsHeldByCurrentThread ()
            FatalError ("Attempt to wait on condition when mutex is not held")
          endIf
          oldIntStat = SetInterruptsTo (DISABLED)
          mutex.Unlock ()
          waitingThreads.AddToEnd (currentThread)
          currentThread.Sleep ()

          -- no lock

          oldIntStat = SetInterruptsTo (oldIntStat)
        endMethod
```

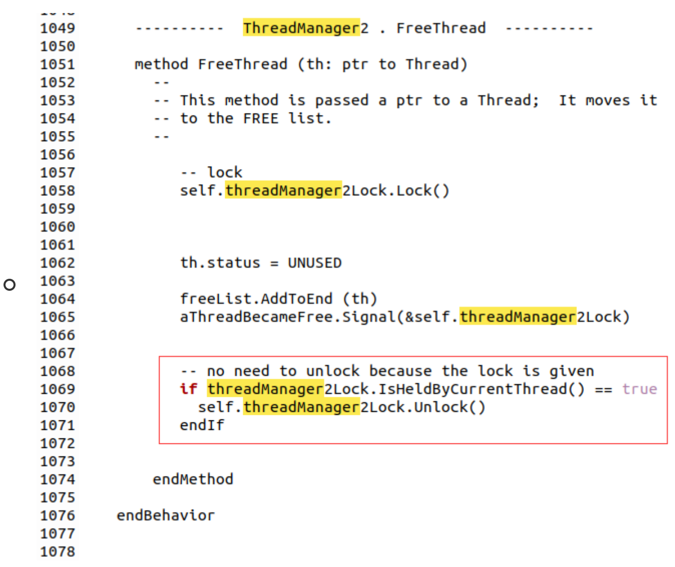
3、测试

为了测试，笔者将lab4中与ThreadManager类与测试有关的所有代码写成ThreadManager2：

- 将其中的mutex换为HoareMutex，condition换为HoareCondition，并将判断条件的while改成if.
- **还需要在以下两处分别添加判断，防止重复上锁或在最后时没有解锁：**

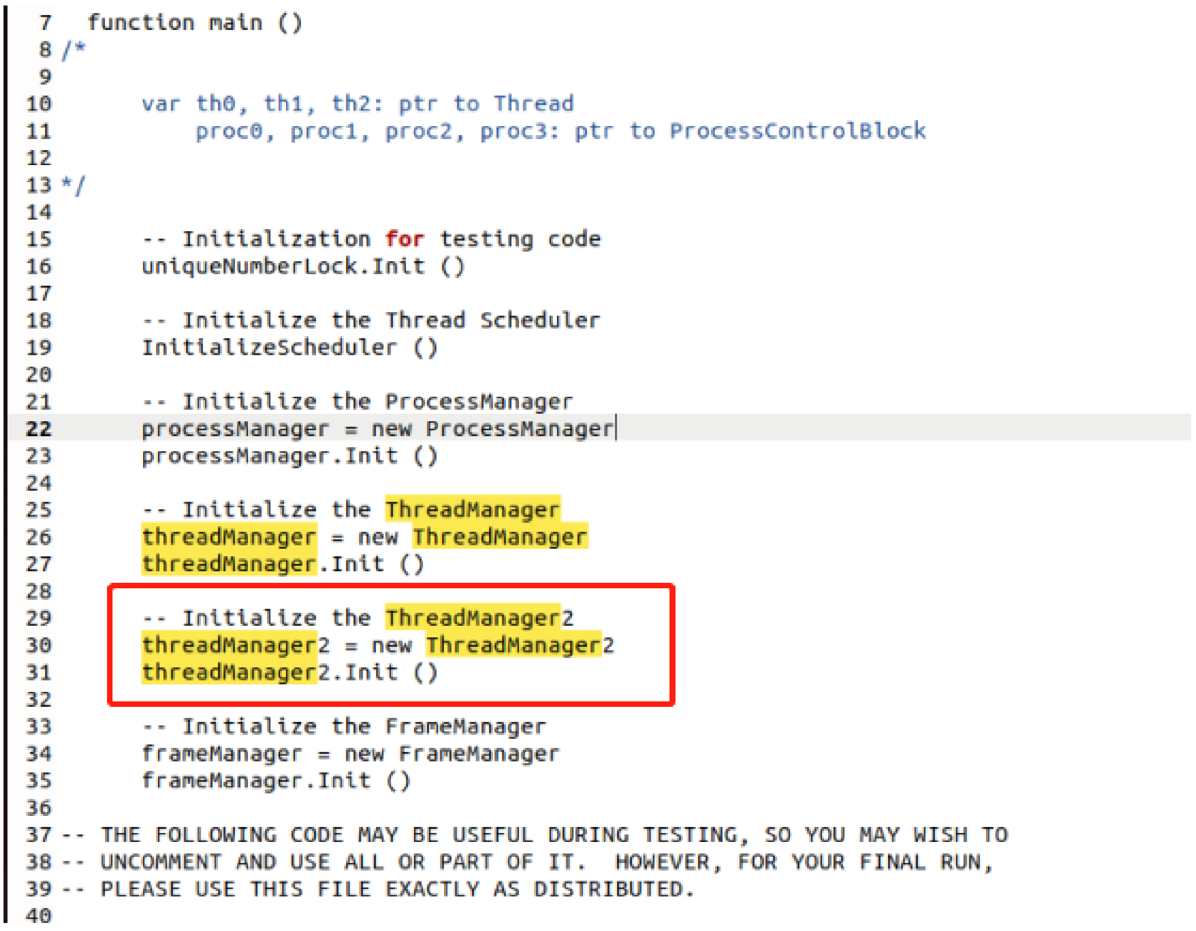


```
      ----------  ThreadManager2 . GetANewThread  ----------

      method GetANewThread () returns ptr to Thread
        --
        -- This method returns a new Thread; it will wait
        -- until one is available.
        --
        var th: ptr to Thread


          -- lock
          if threadManager2Lock.IsHeldByCurrentThread() == false
            self.threadManager2Lock.Lock()
          endIf

          if freeList.IsEmpty() == true
            aThreadBecameFree.Wait(&self.threadManager2Lock)
          endIf

          -- remove and return_ a thread from freelist
          th = freeList.Remove()


          th.status = JUST_CREATED


          -- unlock
          self.threadManager2Lock.Unlock()

          return th
        endMethod
```

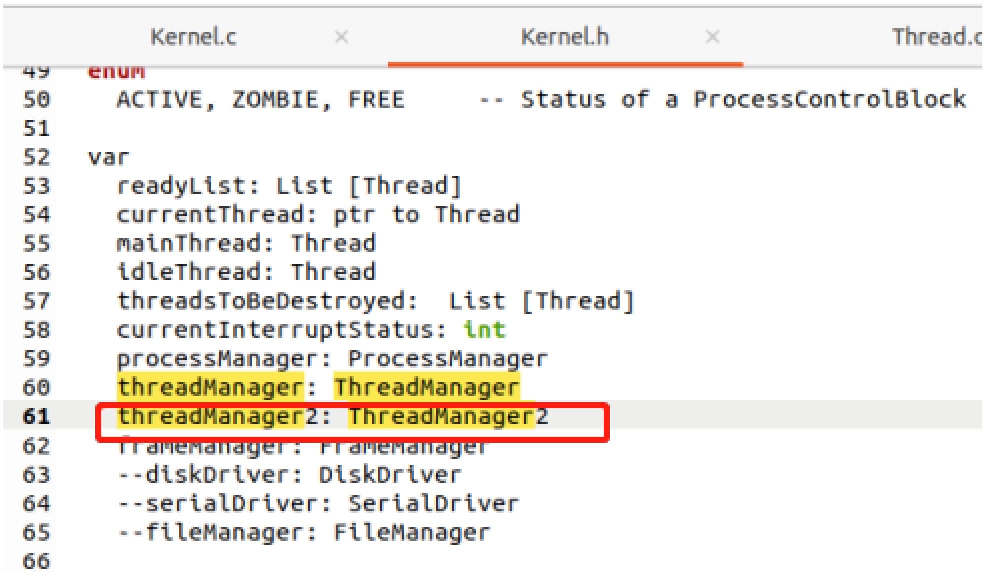


```
  ----------  ThreadManager2 . FreeThread  ----------

  method FreeThread (th: ptr to Thread)
    --
    -- This method is passed a ptr to a Thread;  It moves it
    -- to the FREE list.
    --

      -- lock
      self.threadManager2Lock.Lock()


      th.status = UNUSED

      freeList.AddToEnd (th)
      aThreadBecameFree.Signal(&self.threadManager2Lock)


      -- no need to unlock because the lock is given
      if threadManager2Lock.IsHeldByCurrentThread() == true
        self.threadManager2Lock.Unlock()
      endIf


    endMethod

endBehavior
```

具体代码改动如下所示（所有代码都是新增，不会影响task1~3的执行。除了初始化会导致进程执行时间不同导致输出结果不同）：

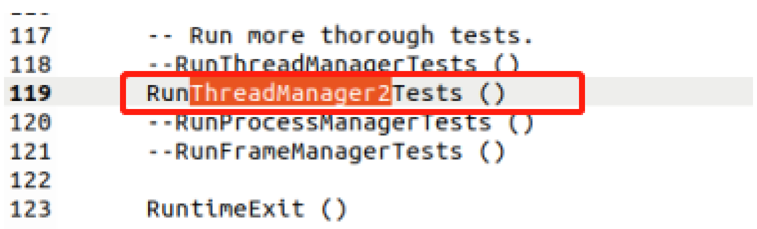


```
function main ()
/*

     var th0, th1, th2: ptr to Thread
         proc0, proc1, proc2, proc3: ptr to ProcessControlBlock

*/

     -- Initialization for testing code
     uniqueNumberLock.Init ()

     -- Initialize the Thread Scheduler
     InitializeScheduler ()

     -- Initialize the ProcessManager
     processManager = new ProcessManager
     processManager.Init ()

     -- Initialize the ThreadManager
     threadManager = new ThreadManager
     threadManager.Init ()

     -- Initialize the ThreadManager2
     threadManager2 = new ThreadManager2
     threadManager2.Init ()

     -- Initialize the FrameManager
     frameManager = new FrameManager
     frameManager.Init ()

-- THE FOLLOWING CODE MAY BE USEFUL DURING TESTING, SO YOU MAY WISH TO
-- UNCOMMENT AND USE ALL OR PART OF IT.  HOWEVER, FOR YOUR FINAL RUN,
-- PLEASE USE THIS FILE EXACTLY AS DISTRIBUTED.
```



Kernel.c × Kernel.h × Thread.c

```
enum
  ACTIVE, ZOMBIE, FREE     -- Status of a ProcessControlBlock

var
  readyList: List [Thread]
  currentThread: ptr to Thread
  mainThread: Thread
  idleThread: Thread
  threadsToBeDestroyed:  List [Thread]
  currentInterruptStatus: int
  processManager: ProcessManager
  threadManager: ThreadManager
  threadManager2: ThreadManager2
  frameManager: FrameManager
  --diskDriver: DiskDriver
  --serialDriver: SerialDriver
  --fileManager: FileManager

```



```
    -- Run more thorough tests.
    --RunThreadManagerTests ()
    RunThreadManager2Tests ()
    --RunProcessManagerTests ()
    --RunFrameManagerTests ()

    RuntimeExit ()
```

```
-----------------------------  RunThreadManager2Tests  ---------------------------------
--
-- This function tests the ThreadManager.  It creates a bunch of threads
-- (NUM_THREADS) and starts each thread running.  Each thread will execute
-- the "TestThreadManager" function.  The main thread will then wait until all
-- the threads complete.  To control this, there is a single Semaphore "allDone".
-- Each TestThreadManager thread signals it and the main thread will wait
-- for NUM-THREAD times, i.e., until all threads have finished.
--
-- Each TestThreadManager does basically this:
--        loop NUMBER_ITERATIONS times
--           call GetANewThread
--           wait
--           call FreeThread
--           wait
--        endLoop
--
  function RunThreadManager2Tests ()
      var i: int
          th: ptr to Thread

      allDone.Init (0)
      freeze.Init (0)
      uniqueNumberLock.Init ()
      nextUnique = 1

```

```
   print ("\n\n*****  THREAD-MANAGER TEST  *****\n\n")

   for i = 1 to NUM_THREADS
     th = alloc Thread
     th.Init ("TestThreadManager2")
     th.Fork (TestThreadManager2, i)
   endFor

   -- Wait for all the testing threads to complete.
   -- (Make sure you see the completion message!)
   for i = 1 to NUM_THREADS
     allDone.Down ()
   endFor

   if GetUniqueNumber (1) != NUM_THREADS * NUMBER_ITERATIONS + 1
     FatalError ("Concurrency control failure (1)")
   endIf
   print ("\n\n***** THREAD-MANAGER TEST COMPLETED SUCCESSFULLY *****\n\n")

  endFunction
```

```
-----------------------------  TestThreadManager2  ---------------------------------
--
-- This function is the main function for a thread which will test the
-- ThreadManager.  It will request and return Thread objects.  First, it
-- grabs a unique number and stuffs it in the Thread.  Later, it makes sure that
-- the number is unchanged.  It could only have changed if some other tester
-- was allowed to access this Thread object before this tester returned it.
--
  function TestThreadManager2 (myID: int)
     var i, j, e: int
         th: ptr to Thread
     -- printIntVar ("Thread started", myID)
     for i = 1 to NUMBER_ITERATIONS
       printInt (myID)
       e = GetUniqueNumber (1)
       th = threadManager2.GetANewThread ()
       th.regs[0] = e
       for j = 1 to WAIT_TIME+i
         currentThread.Yield ()
       endFor
       if e != th.regs[0]
         FatalError ("Concurrency control failure (2)")
       endIf
       printChar ('.')
       threadManager2.FreeThread (th)
       for j = 1 to WAIT_TIME-i
         currentThread.Yield ()
       endFor
     endFor
     allDone.Up ()
     freeze.Down ()
```

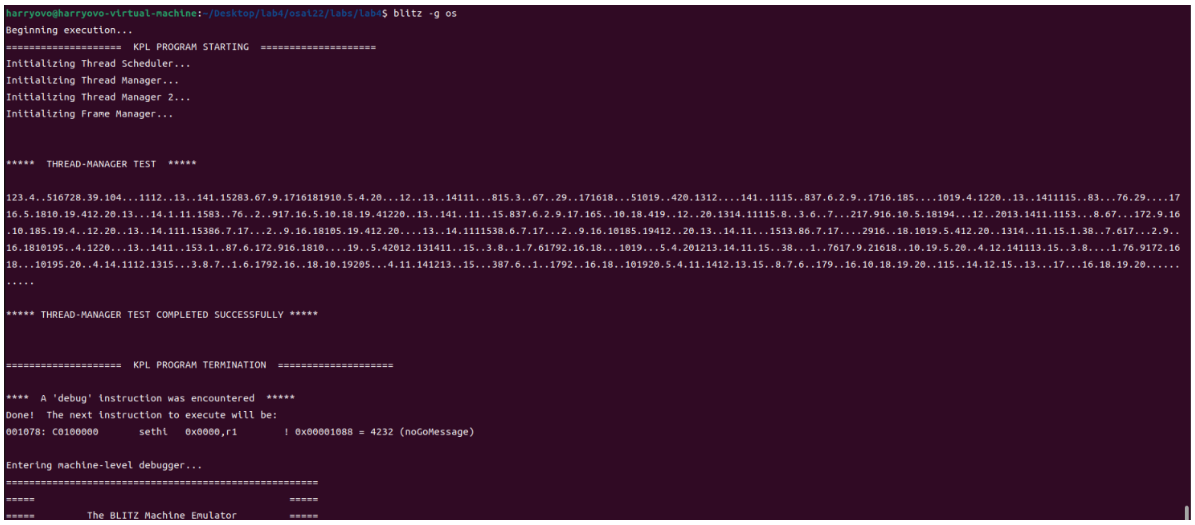
```
    endFunction
```

```
    -----------------------------  ThreadManager2  -------------------------
--------
  --
  --  There is only one instance of this class, created at startup time.
  --
  class ThreadManager2
    superclass Object
    fields
      threadTable: array [MAX_NUMBER_OF_PROCESSES] of Thread
      freeList: List [Thread]

      -- add
      threadManager2Lock: HoareMutex
      aThreadBecameFree: HoareCondition

    methods
      Init ()
      Print ()
      GetANewThread () returns ptr to Thread
      FreeThread (th: ptr to Thread)
  endClass
```

```
-----------------------------  ThreadManager2  -----------------------------
-----

  behavior ThreadManager2

      ----------  ThreadManager2 . Init  ----------

      method Init ()
        --
        -- This method is called once at kernel startup time to initialize
        -- the one and only "ThreadManager2" object.
        --
        var
          i: int



          print ("Initializing Thread Manager 2...\n")

          -- initialize the array of threads
          self.threadTable = new array of Thread {10 of new Thread}

          -- initialize each thread
          self.threadTable[0].Init("0")
          self.threadTable[1].Init("1")
          self.threadTable[2].Init("2")
          self.threadTable[3].Init("3")
          self.threadTable[4].Init("4")
          self.threadTable[5].Init("5")
          self.threadTable[6].Init("6")
          self.threadTable[7].Init("7")
          self.threadTable[8].Init("8")
          self.threadTable[9].Init("9")


          -- initialize freelist
          self.freeList = new List [Thread]

          for i = 0 to 9
            self.threadTable[i].status = UNUSED
            self.freeList.AddToEnd (&threadTable[i])
          endFor


          -- initialize mutex and condition
          self.threadManager2Lock = new HoareMutex
          self.aThreadBecameFree = new HoareCondition

          self.threadManager2Lock.Init()
          self.aThreadBecameFree.Init()

        endMethod

      ----------  ThreadManager2 . Print  ----------

      method Print ()
        --
        -- Print each thread.  Since we look at the freeList, this
        -- routine disables interrupts so the printout will be a
        -- consistent snapshot of things.
        --
        var i, oldStatus: int
          oldStatus = SetInterruptsTo (DISABLED)
          print ("Here is the thread table...\n")
          for i = 0 to MAX_NUMBER_OF_PROCESSES-1
            print ("  ")
            printInt (i)
            print (":")
            ThreadPrintShort (&threadTable[i])
          endFor
          print ("Here is the FREE list of Threads:\n   ")
          freeList.ApplyToEach (PrintObjectAddr)
          nl ()
          oldStatus = SetInterruptsTo (oldStatus)
        endMethod

      ----------  ThreadManager2 . GetANewThread  ----------

      method GetANewThread () returns ptr to Thread
        --
        -- This method returns a new Thread; it will wait
        -- until one is available.
        --
        var th: ptr to Thread

```

```
            -- lock
            if threadManager2Lock.IsHeldByCurrentThread() == false
              self.threadManager2Lock.Lock()
            endIf

            if freeList.IsEmpty() == true
              aThreadBecameFree.Wait(&self.threadManager2Lock)
            endIf

            -- remove and return_ a thread from freelist
            th = freeList.Remove()


            th.status = JUST_CREATED


            -- unlock
            self.threadManager2Lock.Unlock()

            return th
         endMethod

      ----------  ThreadManager2 . FreeThread  ----------

      method FreeThread (th: ptr to Thread)
         --
         -- This method is passed a ptr to a Thread;  It moves it
         -- to the FREE list.
         --

            -- lock
            self.threadManager2Lock.Lock()



            th.status = UNUSED

            freeList.AddToEnd (th)
            aThreadBecameFree.Signal(&self.threadManager2Lock)


            -- no need to unlock because the lock is given
            if threadManager2Lock.IsHeldByCurrentThread() == true
              self.threadManager2Lock.Unlock()
            endIf


         endMethod

    endBehavior
```

结果如下，输出正确：

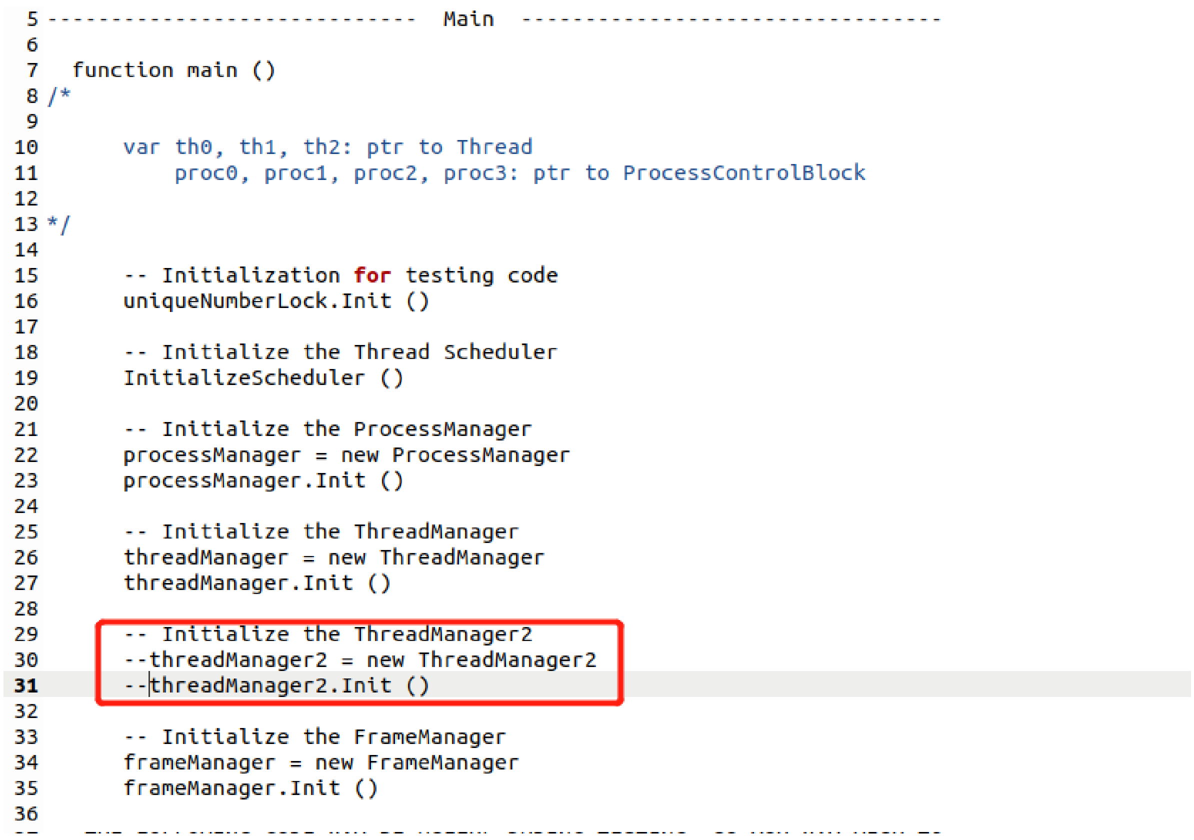


```
harryovo@harryovo-virtual-machine:~/Desktop/lab4/osai22/labs/lab4$ blitz -g os
Beginning execution...
====================  KPL PROGRAM STARTING  ====================
Initializing Thread Scheduler...
Initializing Thread Manager...
Initializing Thread Manager 2...
Initializing Frame Manager...


*****  THREAD-MANAGER TEST  *****

123.4..516728.39.104...1112..13..141.15283.67.9.1716181910.5.4.20...12..13..14111...815.3..67..29..171618...51019..420.1312....141..1115..837.6.2.9..1716.185....1019.4.1220..13..1411115..83...76.29....17
16.5.1810.19.412.20.13...14.1.11.1583..76..2..917.16.5.10.18.19.41220..13..141..11..15.837.6.2.9.17.165..10.18.419..12..20.1314.11115.8..3.6..7...217.916.10.5.18194...12..2013.1411.1153...8.67...172.9.16
.10.185.19.4..12.20..13..14.111.15386.7.17...2..9.16.18105.19.412.20....13..14.1111538.6.7.17...2..9.16.10185.19412..20.13..14.11...1513.86.7.17....2916..18.1019.5.412.20..1314..11.15.1.38..7.617...2.9..
16.1810195..4.1220...13..1411..153.1..87.6.172.916.1810....19..5.42012.131411..15..3.8..1.7.61792.16.18...1019...5.4.201213.14.11.15..38...1..7617.9.21618..10.19.5.20..4.12.141113.15..3.8....1.76.9172.16
18...10195.20..4.14.1112.1315...3.8.7..1.6.1792.16..18.10.19205...4.11.141213..15...387.6..1..1792..16.18..101920.5.4.11.1412.13.15..8.7.6..179..16.10.18.19.20..115..14.12.15..13...17...16.18.19.20......
.....

***** THREAD-MANAGER TEST COMPLETED SUCCESSFULLY *****


====================  KPL PROGRAM TERMINATION  ====================

****  A 'debug' instruction was encountered  *****
Done!  The next instruction to execute will be:
001078: C0100000       sethi   0x0000,r1        ! 0x00001088 = 4232 (noGoMessage)

Entering machine-level debugger...
======================================================
=====                                            =====
=====         The BLITZ Machine Emulator         =====
```

注1：代码修改后由于运行代码的增加，运行时间不一样，输出也会相应变化。

注2：执行task4之前，需要先修改Main.c的下列两处地方；

而在运行task1~3时，需要注释掉threadmanager2的初始化，以得到上面截图中的结果（初始化的增加会影响进程相应的执行时间）：



```
----------------------------  Main  --------------------------------

  function main ()
/*

      var th0, th1, th2: ptr to Thread
          proc0, proc1, proc2, proc3: ptr to ProcessControlBlock

*/

      -- Initialization for testing code
      uniqueNumberLock.Init ()

      -- Initialize the Thread Scheduler
      InitializeScheduler ()

      -- Initialize the ProcessManager
      processManager = new ProcessManager
      processManager.Init ()

      -- Initialize the ThreadManager
      threadManager = new ThreadManager
      threadManager.Init ()

      -- Initialize the ThreadManager2
      --threadManager2 = new ThreadManager2
      --threadManager2.Init ()

      -- Initialize the FrameManager
      frameManager = new FrameManager
      frameManager.Init ()

```

```
    -- Run more thorough tests.
    --RunThreadManagerTests ()
    RunThreadManager2Tests ()
    --RunProcessManagerTests ()
    --RunFrameManagerTests ()

```