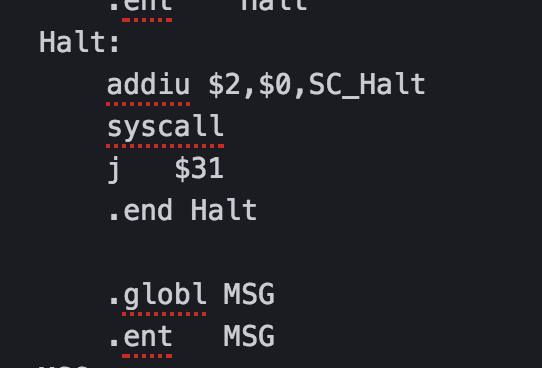
1. ***Trace system call***

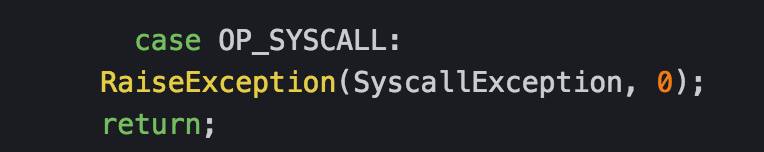
a. Halt(): 最凶狠的system call

以testcase : halt.c為例，當我們include”syscall.h”之後，我們就可以使用syscall中halt()這個function。當這個c檔在compile的時候遇到halt()會轉換成start.S中以下這段組合語言：



如此Mips Machine Simulator跑到這行instruction─ SC\_Halt時會先將這個代碼會存在$2裡面，接下來執行syscall。

接下來看到mipssim.cc，這個檔案會模擬cpu，在Run()這個function中有一個無線迴圈不斷地執行OneInstruction()，而OneInstruction()就是依照PC\_counter去nachos的memory中拿一行instruction出來進行decode，如果出來的opCode符合下面這個case的話：



就會RaiseException()，並傳入SyscallException這個ExceptionType。

RaiseException()這個function是在machine.cc中，首先它會把mode換成SystemMode(也就是kernel mode)，執行ExceptionHandler()並傳入ExceptionType(這邊是SyscallException)，執行完ExceptionHandler()後會將mode換回UserMode。

ExceptionHandler()會在Exception.cc中，首先他會從$2中讀出type(這邊是SC\_Halt)，接下來看傳入的ExceptionType，如果是SyscallException的話就去看type是否符合其中一項case，在這邊符合：



並執行其中的SysHalt()。

SysHalt()在ksyscall.h裡面：



而裡頭就是執行kernel裡的interrupt物件的Halt()，實作細節在interrupt.cc中：



這裡會把kernel delete掉，也就是結束nachos 作業系統的運作。

b. Create(): 檔案之母

1. ***Report***