Homework 1

Operating System, Fall 2024

NachOS System Call

- Calling convention for NachOS
  - r2 – System Call No.
  - r4 – arg1
  - r5 – arg2
  - r6 – arg3
  - r7 – arg4
- Return value (or code) must be put back into r2

NachOS System Call

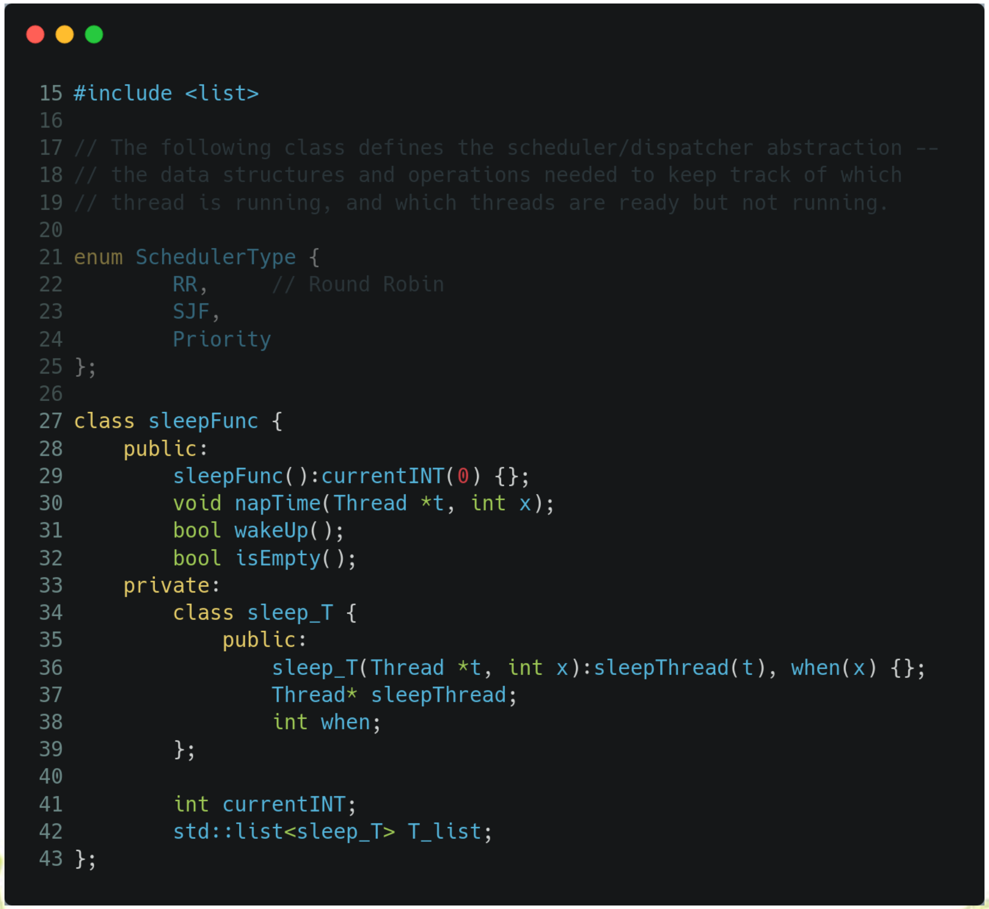
- val = kernel->machine->ReadRegister(4);
  - Read register r4 to val
- kernel->machine->WriteRegister(2, val);
  - Write register r2 from val
- kernel->machine->ReadMem(500, 1, &val);
  - Read from memory address 500, 1 byte of data to val

Lab: Implement SYSCALL Sleep

- Implement system call Sleep(int X)
  - Pause the thread for X microseconds
- Create a test program for Sleep(int X) system call.
- Update the Makefile to include the test program.
- Compile NachOS and execute the test program.

Lab: Implement SYSCALL Sleep

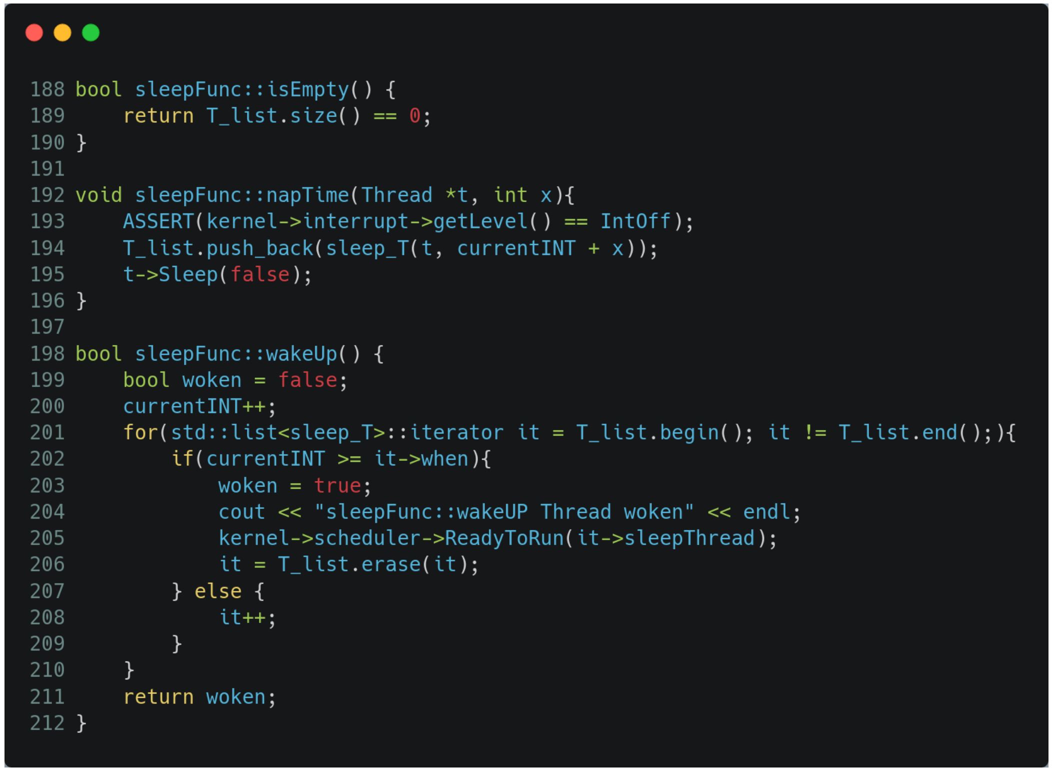
- NachOS/code/threads/scheduler.h



```
#include <list>

// The following class defines the scheduler/dispatcher abstraction --
// the data structures and operations needed to keep track of which
// thread is running, and which threads are ready but not running.

enum SchedulerType {
        RR,     // Round Robin
        SJF,
        Priority
};

class sleepFunc {
    public:
        sleepFunc():currentINT(0) {};
        void napTime(Thread *t, int x);
        bool wakeUp();
        bool isEmpty();
    private:
        class sleep_T {
            public:
                sleep_T(Thread *t, int x):sleepThread(t), when(x) {};
                Thread* sleepThread;
                int when;
        };

        int currentINT;
        std::list<sleep_T> T_list;
};
```

Lab: Implement SYSCALL Sleep

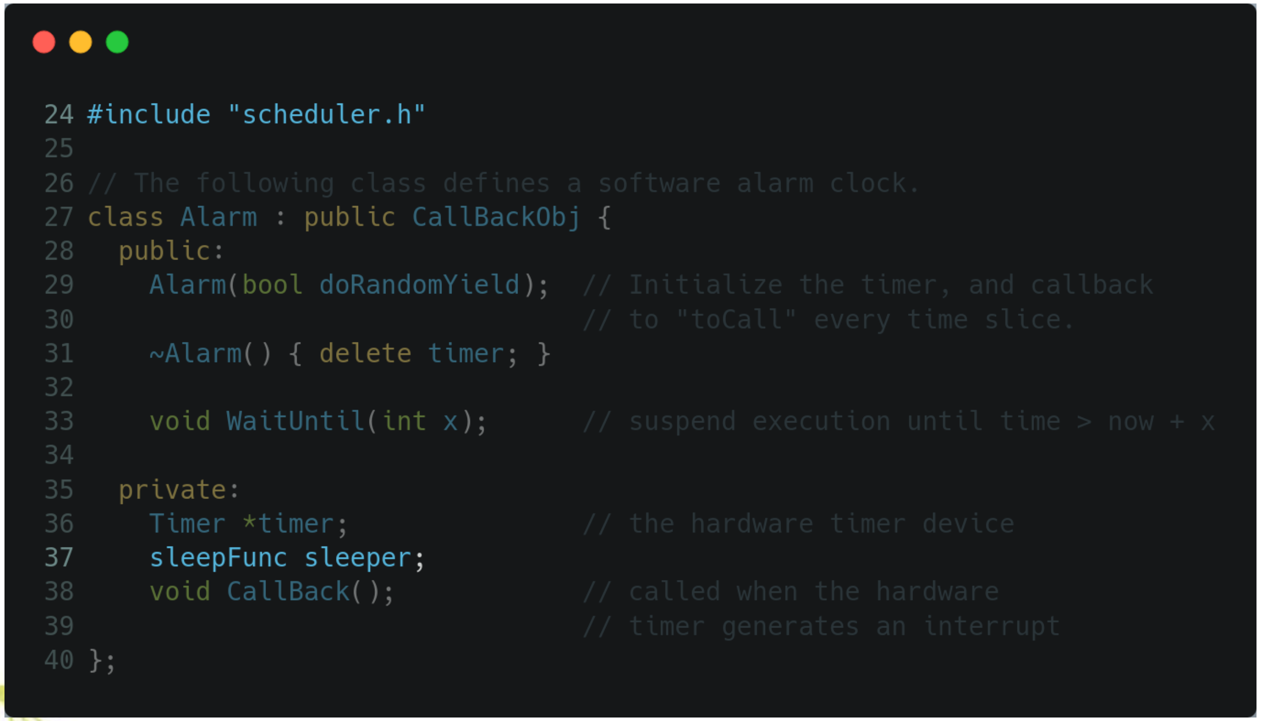
- NachOS/code/threads/scheduler.cc



```
bool sleepFunc::isEmpty() {
    return T_list.size() == 0;
}

void sleepFunc::napTime(Thread *t, int x){
    ASSERT(kernel->interrupt->getLevel() == IntOff);
    T_list.push_back(sleep_T(t, currentINT + x));
    t->Sleep(false);
}

bool sleepFunc::wakeUp() {
    bool woken = false;
    currentINT++;
    for(std::list<sleep_T>::iterator it = T_list.begin(); it != T_list.end();){
        if(currentINT >= it->when){
            woken = true;
            cout << "sleepFunc::wakeUP Thread woken" << endl;
            kernel->scheduler->ReadyToRun(it->sleepThread);
            it = T_list.erase(it);
        } else {
            it++;
        }
    }
    return woken;
}
```

Lab: Implement SYSCALL Sleep

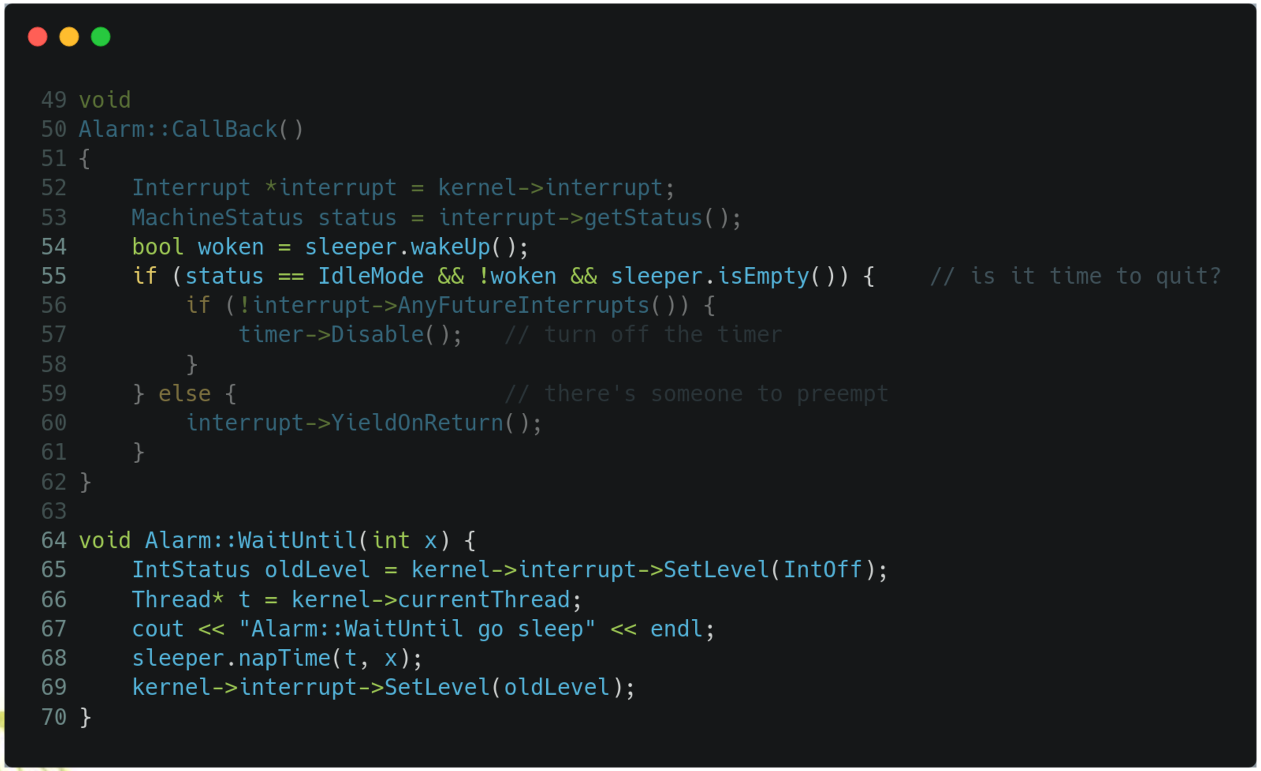
- NachOS/code/threads/alarm.h



```
#include "scheduler.h"

// The following class defines a software alarm clock.
class Alarm : public CallBackObj {
  public:
    Alarm(bool doRandomYield);  // Initialize the timer, and callback
                                // to "toCall" every time slice.
    ~Alarm() { delete timer; }

    void WaitUntil(int x);      // suspend execution until time > now + x

  private:
    Timer *timer;               // the hardware timer device
    sleepFunc sleeper;
    void CallBack();            // called when the hardware
                                // timer generates an interrupt
};
```

Lab: Implement SYSCALL Sleep

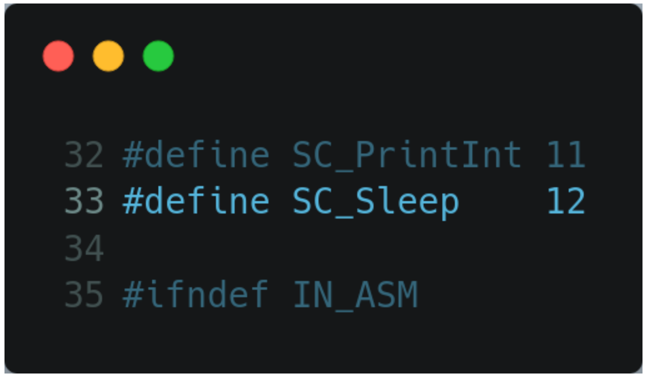
- NachOS/code/threads/alarm.cc



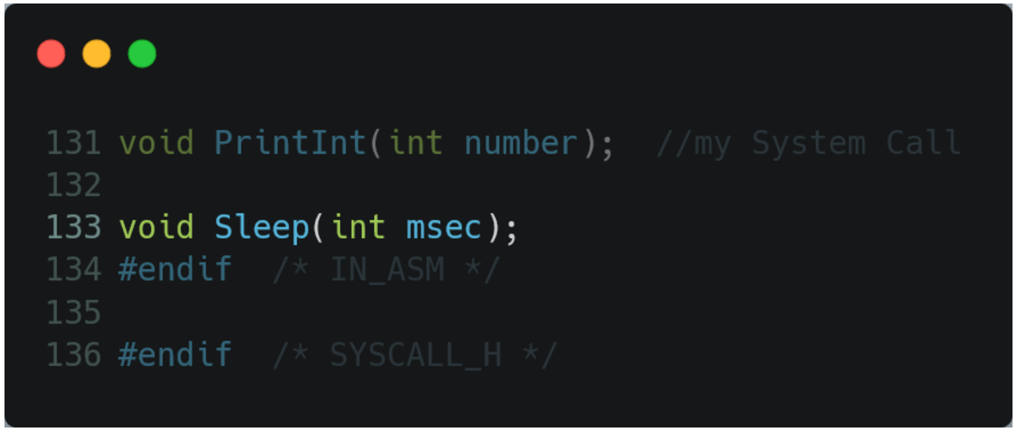
```
void
Alarm::CallBack()
{
    Interrupt *interrupt = kernel->interrupt;
    MachineStatus status = interrupt->getStatus();
    bool woken = sleeper.wakeUp();
    if (status == IdleMode && !woken && sleeper.isEmpty()) {    // is it time to quit?
        if (!interrupt->AnyFutureInterrupts()) {
            timer->Disable();   // turn off the timer
        }
    } else {                    // there's someone to preempt
        interrupt->YieldOnReturn();
    }
}

void Alarm::WaitUntil(int x) {
    IntStatus oldLevel = kernel->interrupt->SetLevel(IntOff);
    Thread* t = kernel->currentThread;
    cout << "Alarm::WaitUntil go sleep" << endl;
    sleeper.napTime(t, x);
    kernel->interrupt->SetLevel(oldLevel);
}
```

Lab: Implement SYSCALL Sleep

- NachOS/code/userprog/syscall.h



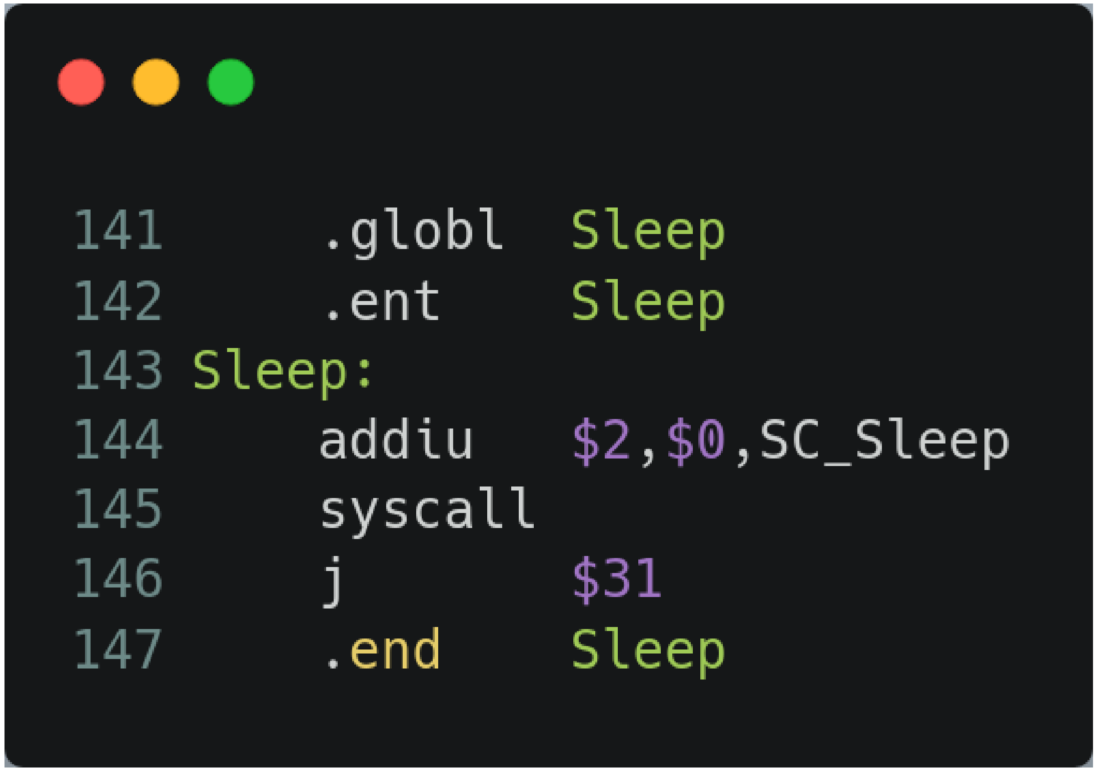
```
#define SC_PrintInt 11
#define SC_Sleep    12

#ifndef IN_ASM
```



```
void PrintInt(int number);  //my System Call

void Sleep(int msec);
#endif  /* IN_ASM */

#endif  /* SYSCALL_H */
```

Lab: Implement SYSCALL Sleep

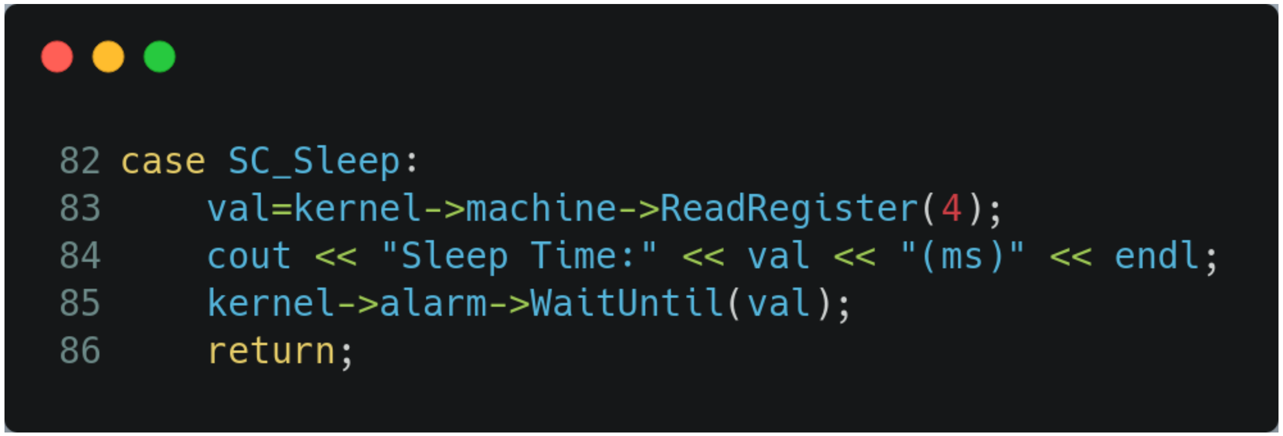
- NachOS/code/test/start.s



```
        .globl  Sleep
        .ent    Sleep
Sleep:
        addiu   $2,$0,SC_Sleep
        syscall
        j       $31
        .end    Sleep
```

Lab: Implement SYSCALL Sleep

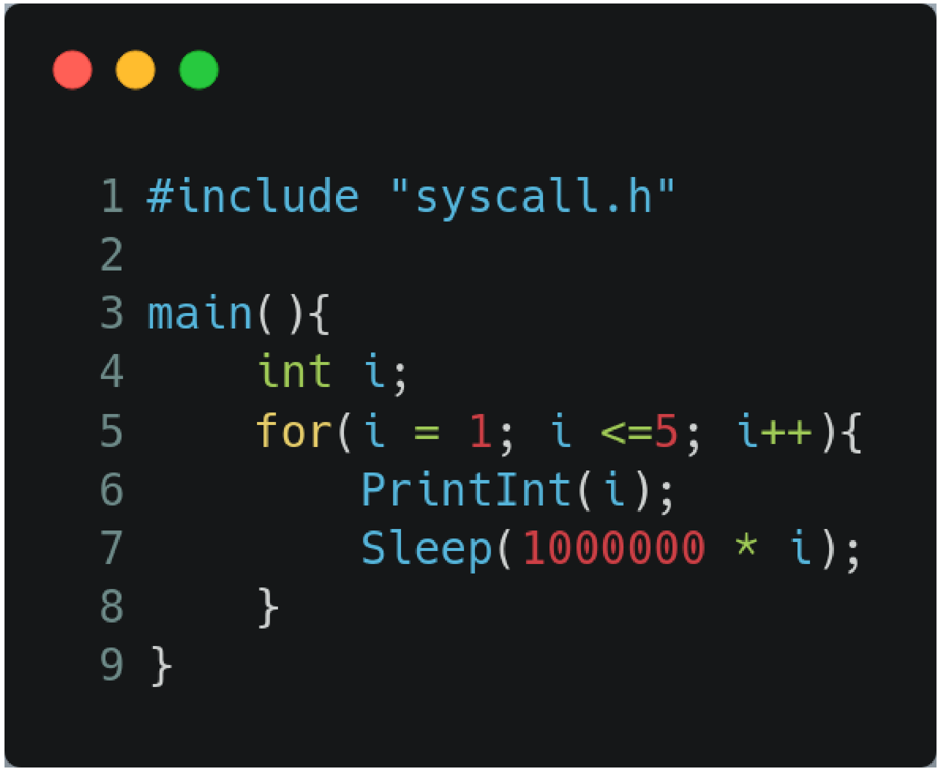
- NachOS/code/userprog/exception.cc



```
case SC_Sleep:
    val=kernel->machine->ReadRegister(4);
    cout << "Sleep Time:" << val << "(ms)" << endl;
    kernel->alarm->WaitUntil(val);
    return;
```

Lab: Implement SYSCALL Sleep

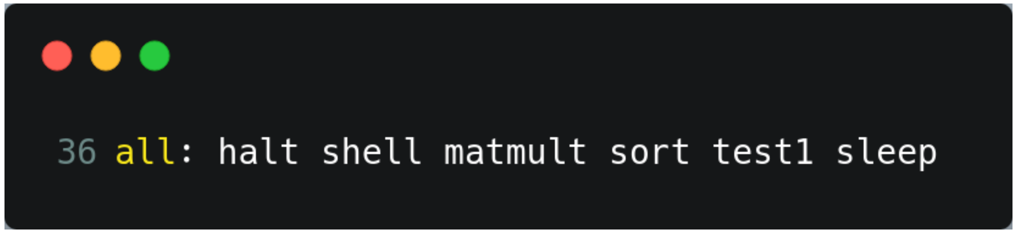
- NachOS/code/test/sleep.c



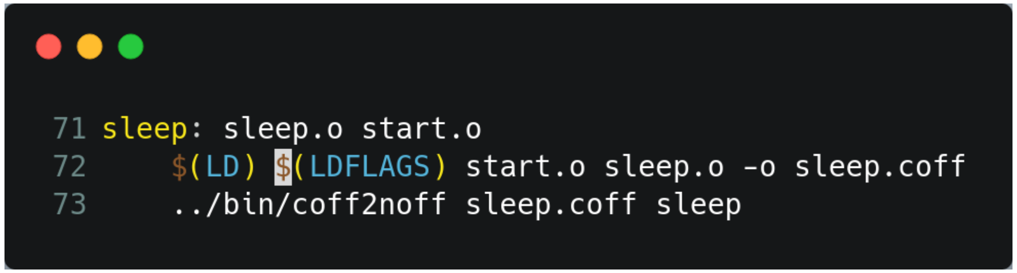
```
#include "syscall.h"

main(){
     int i;
     for(i = 1; i <=5; i++){
          PrintInt(i);
          Sleep(1000000 * i);
     }
}
```

Lab: Implement SYSCALL Sleep

- NachOS/code/test/Makefile



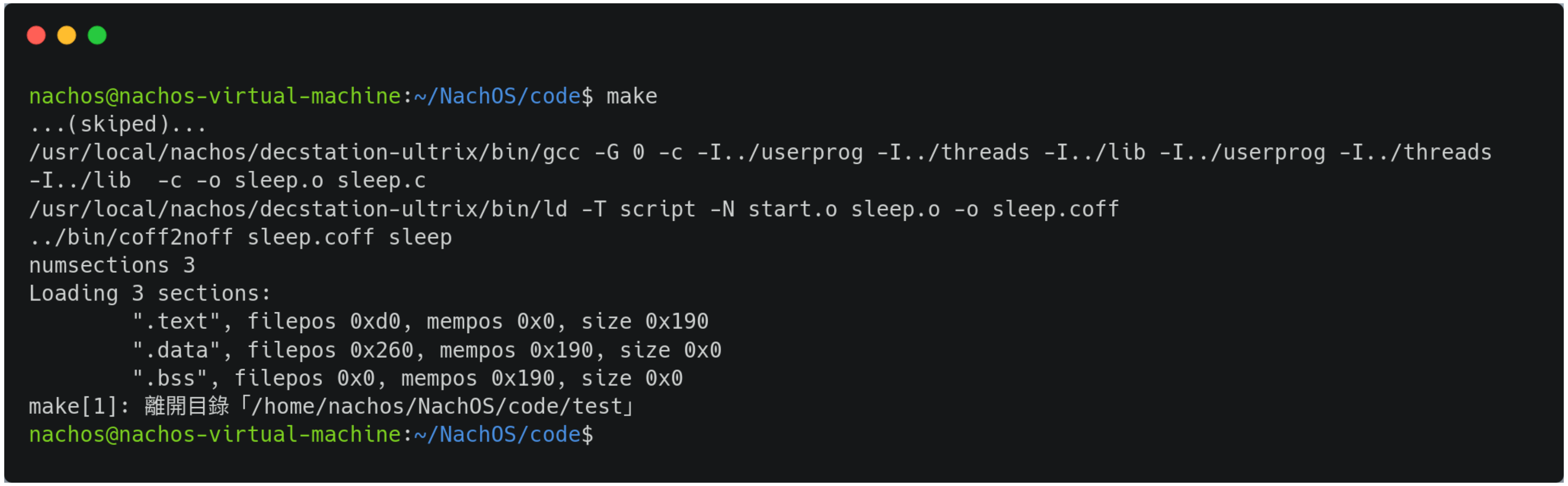
```
all: halt shell matmult sort test1 sleep
```



```
sleep: sleep.o start.o
	$(LD) $(LDFLAGS) start.o sleep.o -o sleep.coff
	../bin/coff2noff sleep.coff sleep
```

Lab: Implement SYSCALL Sleep

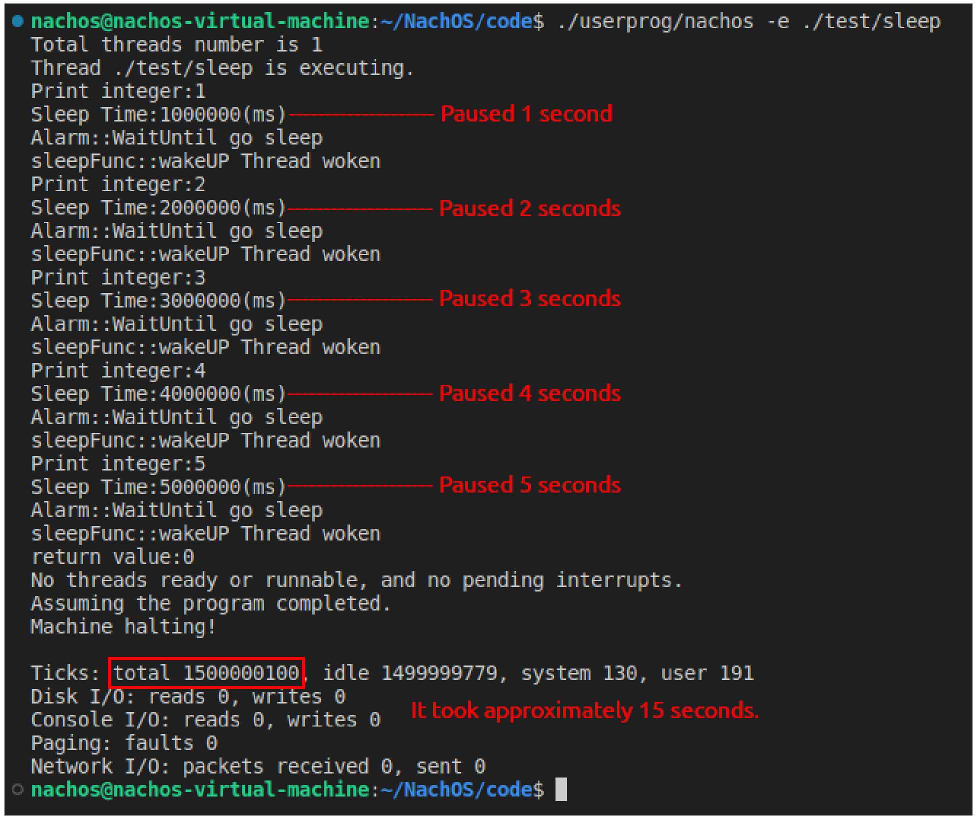
- Recompile NachOS



```
nachos@nachos-virtual-machine:~/NachOS/code$ make
...(skiped)...
/usr/local/nachos/decstation-ultrix/bin/gcc -G 0 -c -I../userprog -I../threads -I../lib -I../userprog -I../threads
-I../lib  -c -o sleep.o sleep.c
/usr/local/nachos/decstation-ultrix/bin/ld -T script -N start.o sleep.o -o sleep.coff
../bin/coff2noff sleep.coff sleep
numsections 3
Loading 3 sections:
        ".text", filepos 0xd0, mempos 0x0, size 0x190
        ".data", filepos 0x260, mempos 0x190, size 0x0
        ".bss", filepos 0x0, mempos 0x190, size 0x0
make[1]: 離開目錄「/home/nachos/NachOS/code/test」
nachos@nachos-virtual-machine:~/NachOS/code$
```

Lab: Implement SYSCALL Sleep

- Run sleep program



```
nachos@nachos-virtual-machine:~/NachOS/code$ ./userprog/nachos -e ./test/sleep
Total threads number is 1
Thread ./test/sleep is executing.
Print integer:1
Sleep Time:1000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
Print integer:2
Sleep Time:2000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
Print integer:3
Sleep Time:3000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
Print integer:4
Sleep Time:4000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
Print integer:5
Sleep Time:5000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 1500000100, idle 1499999779, system 130, user 191
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
nachos@nachos-virtual-machine:~/NachOS/code$
```

HW1 Assignment

- Implement SYSCALL (90%)
  - [Lab1]Sleep (30%)
  - Add/Sub/Mul/Div/Mod (30%)
  - Print (30%)
- Debug Flag (10%)
- Pay attention to letter case

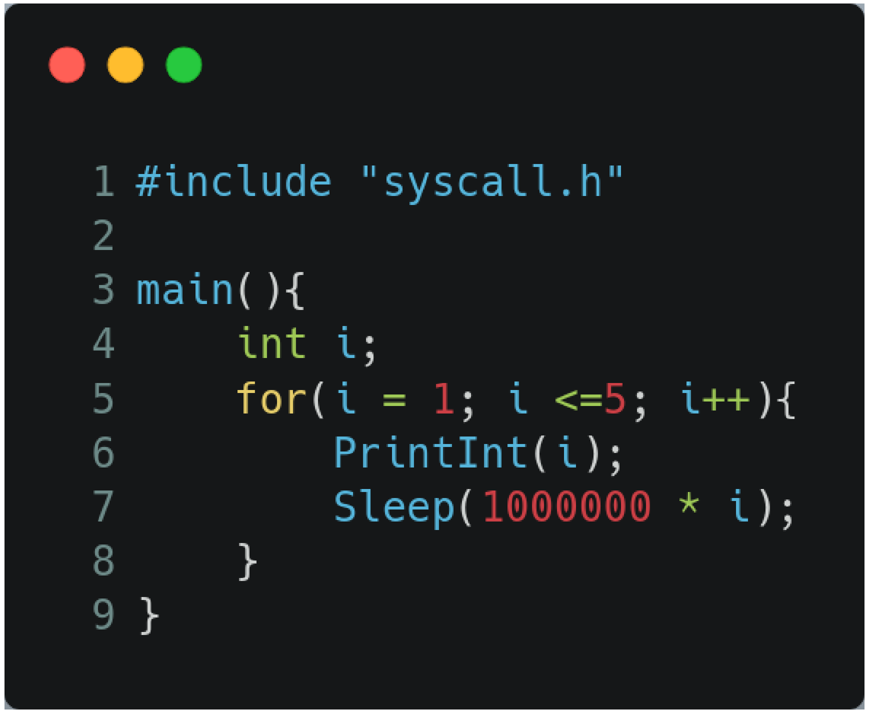
HW1: Implement SYSCALL Sleep

- This system call pause the thread for X microseconds
- A hardware timer generates a CPU interrupt every X microseconds
- The interrupt is triggered by the NachOS Class Alarm

- Important: Avoid long thread sleep periods to prevent potential overflow issues

HW1: Implement SYSCALL Sleep

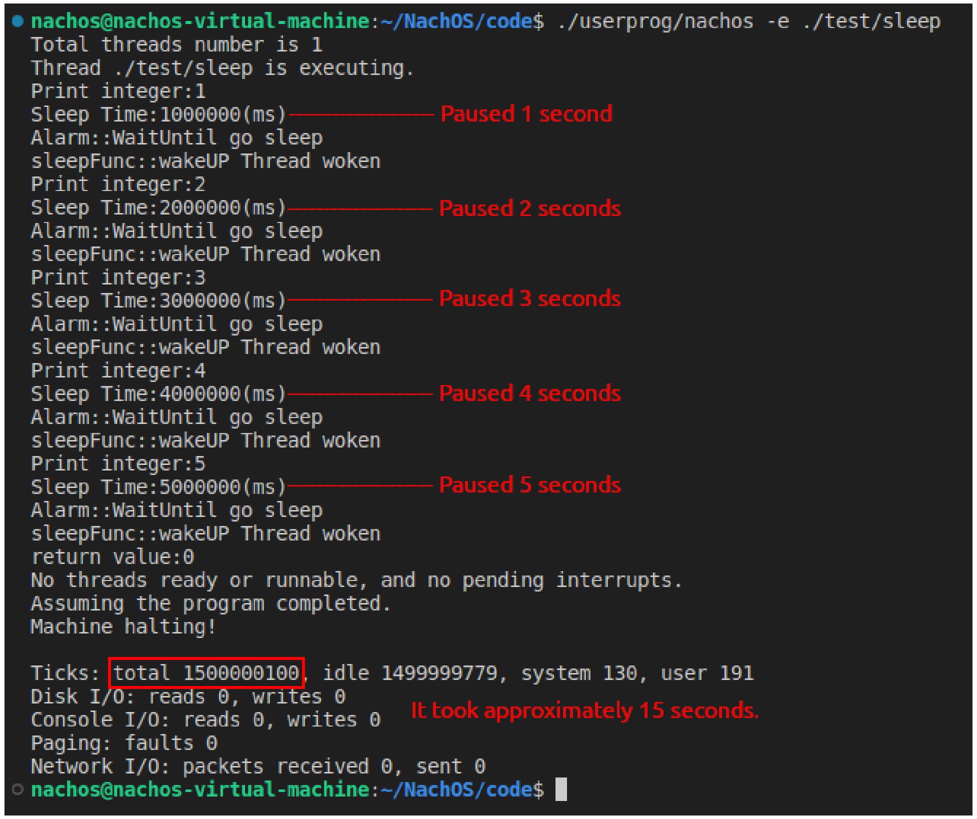
- sleep.c



```c
#include "syscall.h"

main(){
    int i;
    for(i = 1; i <=5; i++){
        PrintInt(i);
        Sleep(1000000 * i);
    }
}
```

HW1: Implement SYSCALL Sleep

- Expected result:



```
nachos@nachos-virtual-machine:~/NachOS/code$ ./userprog/nachos -e ./test/sleep
Total threads number is 1
Thread ./test/sleep is executing.
Print integer:1
Sleep Time:1000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
Print integer:2
Sleep Time:2000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
Print integer:3
Sleep Time:3000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
Print integer:4
Sleep Time:4000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
Print integer:5
Sleep Time:5000000(ms)
Alarm::WaitUntil go sleep
sleepFunc::wakeUP Thread woken
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 1500000100, idle 1499999779, system 130, user 191
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
nachos@nachos-virtual-machine:~/NachOS/code$
```

HW1: SYSCALL Sleep Tips

- Put thread into ready queue after process has waken
  - Tip: `kernel->scheduler->ReadyToRun(thread);`
- You must ensure process won’t be preempted when it runs in `WaitUntil()` function
  - Tip: `kernel->interrupt->SetLevel(IntStatus);`

- You should be able to complete this assignment as we've already provided guidance on implementing the sleep system call during the lab sessions.

HW1: Implement SYSCALL Add/Sub/Mul/Div/Mod

- These system calls perform calculations in the kernel space and then return the results to the user space.
  - Add(op1, op2) - Addition
  - Sub(op1, op2) - Subtraction
  - Mul(op1, op2) – Multiplication
  - Div(op1, op2) – Division
  - Mod(op1, op2) – Modulus

HW1: Implement SYSCALL Add/Sub/Mul/Div/Mod

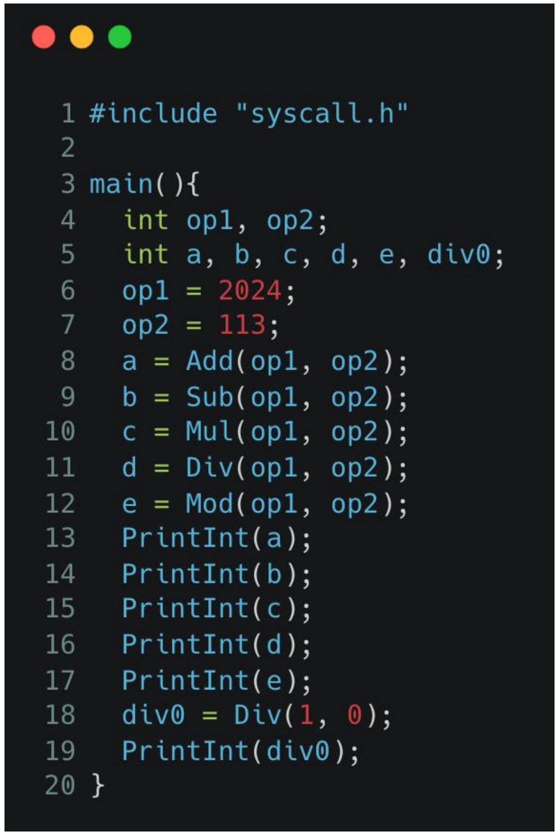
- In the event of a divide-by-zero error
  - Avoid NachOS core dumping with Floating Point Exception
  - Print the error message
  - Return the last eight digits of the student ID
- For example:
  - My student ID is M11215085

  -
    ```
    div0 = Div(1, 0);
    PrintInt(div0);
    ```
- Should output:

  -
    ```
    Error: Divide by zero
    Print Integer:11215085
    ```

HW1: Implement SYSCALL Add/Sub/Mul/Div/Mod

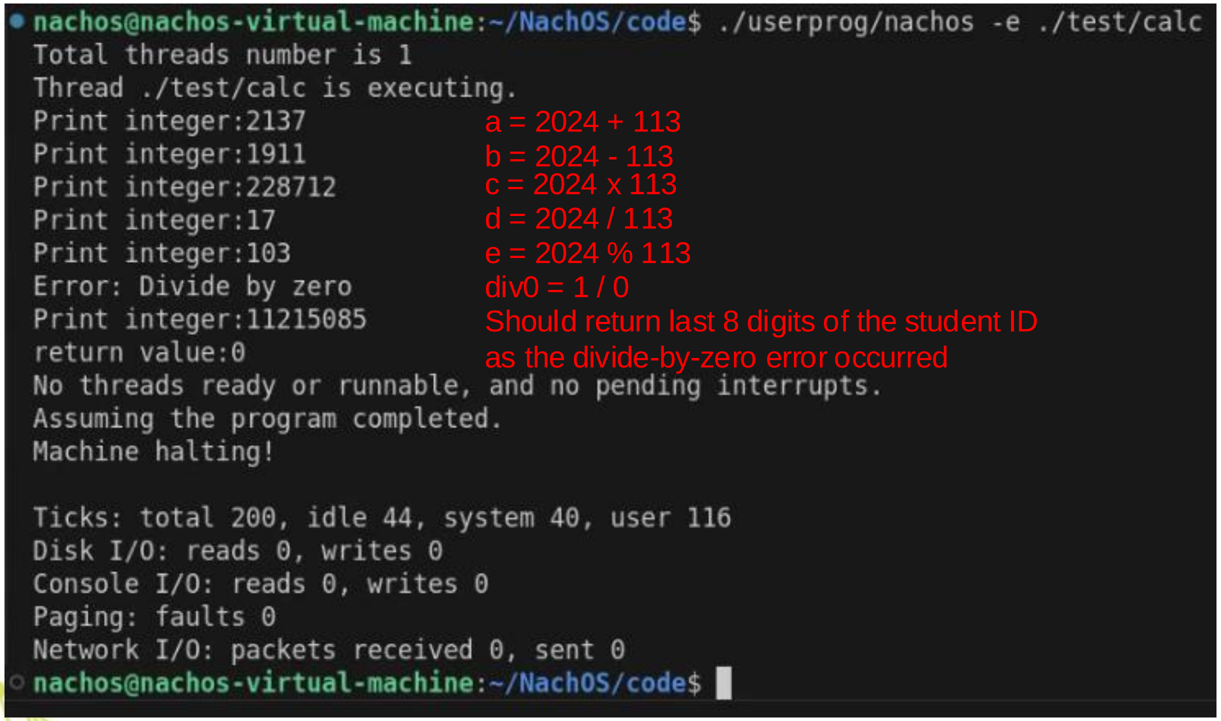
- calc.c



```c
#include "syscall.h"

main(){
  int op1, op2;
  int a, b, c, d, e, div0;
  op1 = 2024;
  op2 = 113;
  a = Add(op1, op2);
  b = Sub(op1, op2);
  c = Mul(op1, op2);
  d = Div(op1, op2);
  e = Mod(op1, op2);
  PrintInt(a);
  PrintInt(b);
  PrintInt(c);
  PrintInt(d);
  PrintInt(e);
  div0 = Div(1, 0);
  PrintInt(div0);
}
```

HW1: Implement SYSCALL Add/Sub/Mul/Div/Mod

- Excepted Result



```
nachos@nachos-virtual-machine:~/NachOS/code$ ./userprog/nachos -e ./test/calc
Total threads number is 1
Thread ./test/calc is executing.
Print integer:2137              a = 2024 + 113
Print integer:1911              b = 2024 - 113
Print integer:228712            c = 2024 x 113
Print integer:17                d = 2024 / 113
Print integer:103               e = 2024 % 113
Error: Divide by zero           div0 = 1 / 0
Print integer:11215085          Should return last 8 digits of the student ID
return value:0                  as the divide-by-zero error occurred
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 200, idle 44, system 40, user 116
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
nachos@nachos-virtual-machine:~/NachOS/code$
```

HW1: SYSCALL Add/Sub/Mul/Div/Mod Tips

- These calculations should be performed in kernel space when a SyscallException occurs.
  - Tip: NachOS/code/userprog/exception.cc
- If you're able to implement one of the system call, you should be able to finish the rest of them.
- For system call implementation help, check the specified chapters in the NachOS Tutorial.
  - Adding User Program
  - NachOS System Call
  - Example: Implement SYSCALL

HW1: Implement SYSCALL Print

- This system call prints strings to the console
- The return value should be the number of characters in that string. (excluding \0 in the end)
- Pay attention to letter case
- Output Format:
  - [StudentID_Print](Strings)
  - For example:[M11215085_Print]Hello NachOS2024

HW1: Implement SYSCALL Print

- Everyone has two landmine characters
  - (The last two digits of the student ID) % 26 + ‘A’ and ‘a’
- For example:
  - My student ID is M11215085
  - 85 % 26 = 7
  - My landmine character is h and H

0	1	2	3	4	5	6	7
A	B	C	D	E	F	G	H
a	b	c	d	e	f	g	h

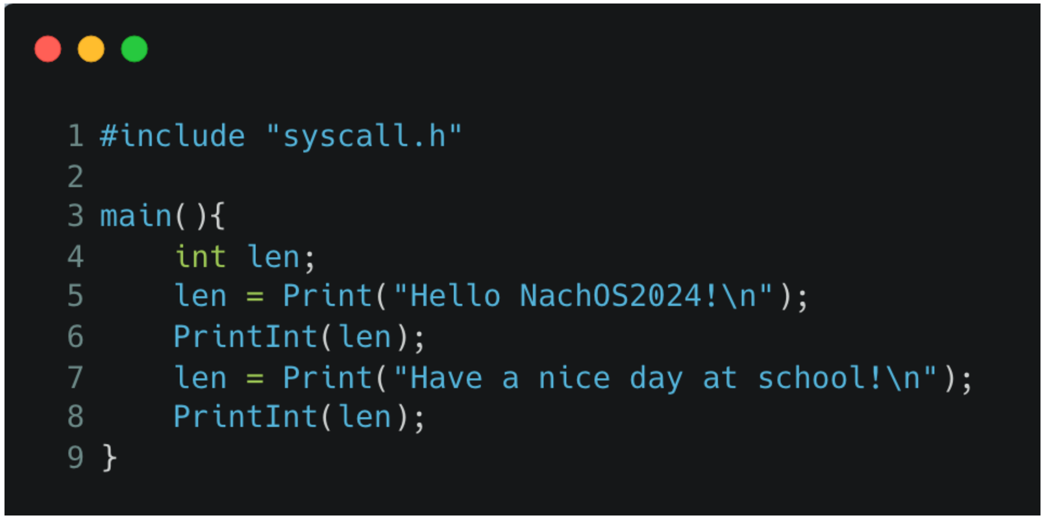
CONNECTIVITY LAB

HW1: Implement SYSCALL Print

- If the input character is a mine character when calling Print(), please do not output it. Instead, output * (Which indicates the boom symbol)
- For example:
  - Print("ABCDEFGHabcdefgh");
- Should output:
  - [M11215085_Print]ABCDEFG*abcdefg*

HW1: Implement SYSCALL Print

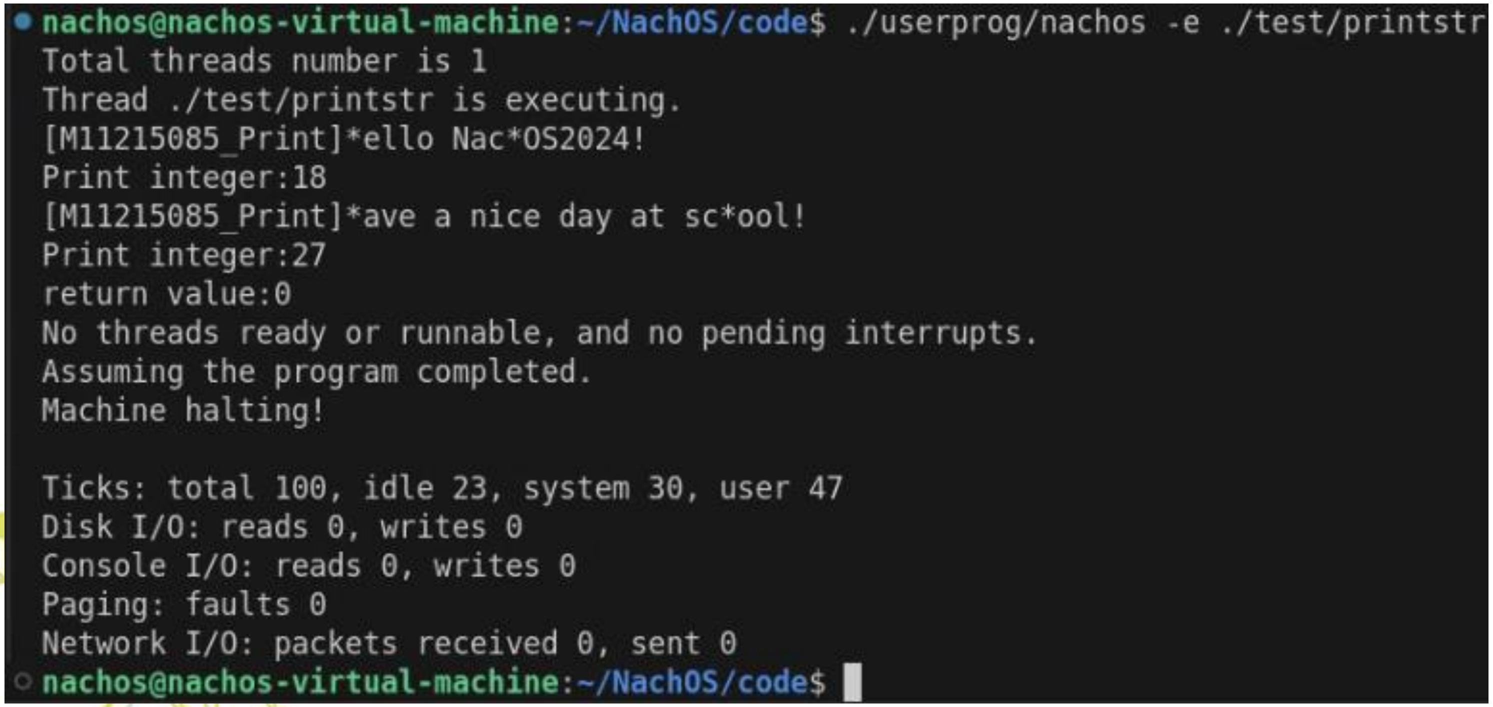
- printstr.c



```c
#include "syscall.h"

main(){
    int len;
    len = Print("Hello NachOS2024!\n");
    PrintInt(len);
    len = Print("Have a nice day at school!\n");
    PrintInt(len);
}
```

HW1: Implement SYSCALL Print

- Expected Result



```
nachos@nachos-virtual-machine:~/NachOS/code$ ./userprog/nachos -e ./test/printstr
Total threads number is 1
Thread ./test/printstr is executing.
[M11215085_Print]*ello Nac*OS2024!
Print integer:18
[M11215085_Print]*ave a nice day at sc*ool!
Print integer:27
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 100, idle 23, system 30, user 47
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
nachos@nachos-virtual-machine:~/NachOS/code$
```

HW1: SYSCALL Print Tips

- Printing string to console should be performed in kernel space when a SyscallException occurs.
  - Tip: NachOS/code/userprog/exception.cc
- You can identify the end of the string by detecting the \0 character.
- For reading memory content, you can use Machine::ReadMem()
  - Tip: kernel->machine->ReadMem(int addr, int size, int* value);

HW1: SYSCALL Print Tips

- For system call implementation help, check the specified chapters in the NachOS Tutorial.
  - Adding User Program
  - NachOS System Call
  - Example: Implement SYSCALL

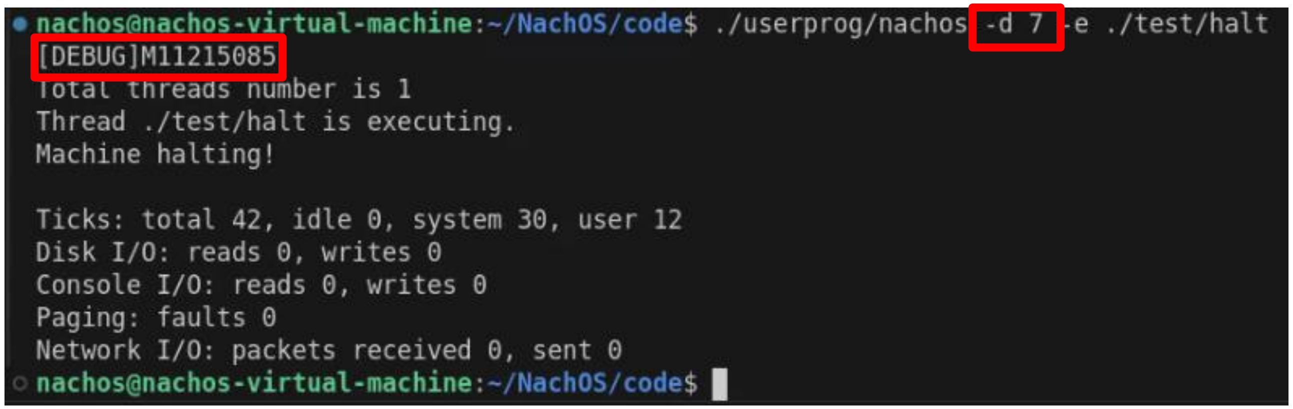
HW1: Debug Flag

- Use the last 2 digits of your student ID % 26 % 10 as your debug flag character
- When the thread runs, print [DEBUG] StudentID
  - For example: My student ID is M11215085
  - 85 % 26 % 10 = 7
  - Use `./userprog/nachos -d 7 -e ../test/test1` to print [DEBUG] M11215085

CONNECTIVITY LAB

HW1: Debug Flag

- Expected Result



```
nachos@nachos-virtual-machine:~/NachOS/code$ ./userprog/nachos -d 7 -e ./test/halt
[DEBUG]M11215085
Total threads number is 1
Thread ./test/halt is executing.
Machine halting!

Ticks: total 42, idle 0, system 30, user 12
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
nachos@nachos-virtual-machine:~/NachOS/code$
```

HW1: Debug Flag Tips

- For debug flag implementation help, check the specified chapters in the NachOS Tutorial.
  - Debug Flag

Hand in Source Code & Report

- **Deadline:**
  - 2024/10/1 23:55
- **Source Code**
  1. Perform a cleanup of build files using make clean
  2. Compress the NachOS source code as StudentID_HW1_src.tar.gz
     - for example, M11215085_HW1_src.tar.gz
  3. Upload the compressed source code to Moodle

Hand in Source Code & Report

- **Report**
  - Your report should include the following:
  1. The approach you took to implement the problem in NachOS
  2. Include essential code snippets and comments for the implementation
  3. Experimental results, including screenshots
  - Save the report as StudentID_HW1_report.pdf
    - For instance, M11215085_HW1_report.pdf
  - Upload the report to Moodle