/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INITIAL.C \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Initial.c is our nucleus initialization or entry point for Kaya,

which takes control when our machine is turned on. It has an initial call of the function main().

In this .c file, we initialize global variables, and physical memory locations.

In addition, it provides the core functionality of basic Operating System that includes global

variables and data structures that manage the flow of our processes (readyQue, procCount, semdTable,

currentProcess, softBlockCount, etc.).

==== Written By Solomon G and Edwin Cervantes ==== \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* #INCLUDE MODULES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* h files to include \*/

#include "../h/const.h"

#include "../h/types.h"

/\* e files to include \*/

#include "../e/pcb.e"

#include "../e/asl.e"

#include "../e/initial.e"

#include "../e/interrupts.e"

#include "../e/exceptions.e"

#include "../e/scheduler.e"

/\* include the µmps2 library \*/

#include "/usr/local/include/umps2/umps/libumps.e"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GLOBAL VARIABLES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* current process count \*/

int processCount;

/\* soft blocked count \*/

int softBlockCount;

/\* current process \*/

pcb\_PTR currentProcess;

/\* ready queue processes \*/

pcb\_PTR readyQueue;

/\* sem table for 49 devices \*/

int semdTable[SEMALLOC];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Importing test file to get the mem address of the function in mikey's p2test file. \*/

extern void test();

/\* Parameters: N/A\*/

/\* Returns: N/A \*/

int main() {

/\* ==== LOCAL VARIABLES ==== \*/

int i; /\* Used to initialize semdTable \*/

unsigned int RAMTOP; /\* Max RAM \*/

devregarea\_PTR registerBus; /\* defining the rdev register area \*/

state\_PTR newState; /\* State \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

registerBus = (devregarea\_PTR) RAMBASEADDR;

RAMTOP = (registerBus->rambase) + (registerBus->ramsize);

/\* ==== INITIALIZE SEMPAHORES FOR SYNCHRONIZATION ==== \*/

/\* with a starting address of 0 \*/

for(i = 0; i < SEMALLOC; i++){

semdTable[i] = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* This is the syscall area \*/

newState = (state\_PTR) SYSCALLNEWAREA;

newState->s\_status = ALLOFF;

newState->s\_sp = RAMTOP;

newState->s\_pc = (memaddr) syscallHandler;

newState->s\_t9 = (memaddr) syscallHandler;

/\* This is the progtrap area \*/

newState = (state\_PTR) PRGMTRAPNEWAREA;

newState->s\_status = ALLOFF;

newState->s\_sp = RAMTOP;

newState->s\_pc = (memaddr) progTrapHandler;

newState->s\_t9 = (memaddr) progTrapHandler;

/\* This is the tblmgmt area \*/

newState = (state\_PTR) TBLMGMTNEWAREA;

newState->s\_status = ALLOFF;

newState->s\_sp = RAMTOP;

newState->s\_pc = (memaddr) tlbHandler;

newState->s\_t9 = (memaddr) tlbHandler;

/\* This is the interrupt area \*/

newState = (state\_PTR) INTERRUPTNEWAREA;

newState->s\_status = ALLOFF;

newState->s\_sp = RAMTOP;

newState->s\_pc = (memaddr) interruptHandler;

newState->s\_t9 = (memaddr) interruptHandler;

/\* initialize the data structures (PCB & ASL) \*/

initPcbs();

initASL();

/\* ==== INITIALIZE VARIABLES ==== \*/

readyQueue = mkEmptyProcQ();

currentProcess = NULL;

processCount = 0;

softBlockCount = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Set up \*/

currentProcess = allocPcb();

processCount++;

currentProcess->p\_state.s\_sp = (RAMTOP - FRAMESIZE);

currentProcess->p\_state.s\_pc = (memaddr) test;

currentProcess->p\_state.s\_t9 = (memaddr) test;

currentProcess->p\_state.s\_status = (ALLOFF | INTERRUPTSON | IM | TE);

/\* insert the newly process into ready queue \*/

insertProcQ(&(readyQueue), currentProcess);

/\* the new process is in the queue \*/

currentProcess = NULL;

/\* load an interval time \*/

LDIT(INTERVALTIME);

/\* Call the scheduler \*/

scheduler();

/\* We get a warning if we don't return a value \*/

return -1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF INITIAL.C \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/