 Zengtai Qi (zxq150130)

Project 1 Summary

CS 5348.001 - Operating Systems Concepts

I. Project Purpose

1. How multiple processes can communicate and cooperate.

   In C programming language, fork() creates a new process referred to as child process by duplicating the calling process. And pipe() creates a unidirectional data channel that can be used for interprocess communication. One process can write data into one end of the pipe and the data will go through the pipe reach the other end of the pipe so that the other process can read this data from the other end of the pipe.

2. Understand processor interaction with main memory.

   The memory isn't aware of what to do with instructions, memory only supports two operation that are read and write. Processor tell memory to read the data in a specific address and the memory give the data back to processor. It's processor that handle the instructions and then simply tell memory whether write some data into a address or fetch some data from a address back in order to complete the instruction behavior.

3. Understand processor instruction behavior.

   Processor instructions are kind of machine language. A given instruction is typically data handling and memory operation or arithmetic and logic operation or control flow operation. Basic instructions are all simple operations. Executing instructions step by step can do a meaningful task.

4. Understand role of registers.

   A register is a small amount of storage available as part of a processor.

   PC is program counter which stores the address of the instruction being executed at the current time. As each instruction gets fetched, PC increases by 1.

   SP is stack pointer which is used to keep track of a stack. SP always stores the address of current stack top, each push operation makes SP increase by 1, and each pop operation makes SP decrease by 1.

   IR is instruction register which stores the instruction currently being executed. Each instruction to be executed is loaded into IR. The value in IR determines what the processor will do which may take several steps.

   AC is accumulator in which intermediate arithmetic and logic results are stored.

5. Understand two forms of interrupts: the timer and system call.

   Both of two forms of interrupts will push the current state of the processor into the stack and then switch PC to the address of the interrupt handler. When a interrupt end, processor pop and restore the state before the interrupt.

   Timer interrupt happens every X instructions have been executed. Timer interrupt handler's exact job depends on the OS architecture.

   System call happens when the processor execute a specific instruction.

6. Understand memory protection.

   Memory protection is a way to control memory access rights on a computer. It prevents a process from accessing memory that is not allocated to the process. An attempt to access unowned memory results in a hardware fault.

II. How the project was implemented

1. Multiple processes and communicating.

   Firstly I created two pipe, one is for send fetch request from processor to memory, and the other one is for send response back from memory. Then I use fork() to create two processes, the parent process acts as a processor and child process acts as the main memory.

2. Memory.

   Before the memory start responding to processor's request, I read the program file into memory line by line. I use atoi() to parse number from every line. If the line begins with a dot, jump to the address of the value of this line and continue reading file.

   After memory completes reading file, memory reads the processor's request and begins working.

   In each request from processor, there is a operationFlag, which indicates the memory to read or write or terminate.

   If the operationFlag indicates reading, memory will wait for one parameter which is the address of data and send the data back.

   If the operationFlag indicates writing, memory will wait for two parameters which are address and data the processor want to store.

3. Processor.

   I declared the six registers and a modeFlag at the beginning of the processor process.

   The processor always firstly send a read request to memory to fetch the instruction into IR according to PC value. Then there is a switch block to determine what processor will do in the next several steps according to the value of IR. Processor will repeatedly fetch a instruction and do instruction behavior until the processor fetches the 50 instruction.

   For example: instruction 23, push return address onto stack, jump to the address.

   When processor fetches the 23 instruction, processor will send a write request to memory, writing (PC + 1) to the address of SP value. Then move SP to SP - 1. At last send a read request to memory to fetch the next value in the program and set the PC to this value completing the 23 instruction.

4. Mode Flag.

   Mode flag indicates which mode is the processor under. 0 indicates user mode and 1 indicates kernel mode. Mode flag is important in my program, because the processor will check the mode in many circumstance.

   When the processor is about to run an interrupt handler, it checks mode flag because if processor is already under a interrupt, it will ignore the nested interrupt.

   And timer decrement only happens under user mode.

5. Timer Interrupt.

There is an int variable in the processor, it's initialize to the value given in the arguments. Each time processor executes an instruction under user mode, the timer variable will decrease by 1. And before processor fetches an instruction, it will check the timer. If timeout, processor will execute the timer interrupt handler.

III. Personal experience in doing the project

1. I know about Java and have never touched C before. However, I choose C in this project, because I want to push myself to get familiar with C.

   Accomplishing this project, I do know more about C, especially the fork(), pipe() and pointers which Java doesn't have. Thought many difference between C and Java, there are more commons, so I didn't feel that hard writing C program.

   Also I learned how to write code with VIM in terminal, how to compile and run program in terminal, and how to take advantage of the remote server.

2. I got deeper understanding with the how a processor generally works. All the high-level programming languages, they will be interpreted into low-level languages program like instructions finally.

   What processors can do are really simple, just easy data handle, easy arithmetic and logic operation and easy control flow operation. But when many instructions are executed in a proper way, processors can also do a complex job.

3. I also got deeper understanding with the role memory acts in a computer. I thought memory just stores user's data before. Now I'm aware that memory stores almost everything of a operating system. Memory stores data, arithmetic values, logic values, address, state of a process and so on.