2020

OPERATING SYSTEMS

HOMEWORK 3 REPORT

AHMET YUŞA TELLİ

151044092

In this we have 3 different SPIMOS_GTU files.

SPIMOS_GTU_1:

In this file, we load 4 different programs and run using Round Robin.

At the beginning, we define our programs' program counters and flags.

Firstly, we add programs to memory one by one, then we save their program counters. After finish this adding, we start to run programs using PC's. We start who is first.

SPIMOS_GTU_2:

In this file, we choose one program and run 5 times. In 29. syscall we generate a random number and select a program. Then sent its name to the spimos_gtu_2 file.

We take file name from cpp. Then we load program to memory and save their program counters.

SPIMOS_GTU_3:

In this file, we choose 3 different programs using 25. syscall. Then we compare which programs came to spimos_gtu file. When we are checking, we load this file. We save three program counters for each program.

CONTEXT SWITCH:

I want to explain how I did "context switching" using spimos_gtu_1.

First, we load "Collatz.asm" file and save its program counter. Then load "LinearSearch.asm" with same way. Then "BinarySearch.asm" and "Palindrome.asm" files.

We take all program counters, now we load to memory our programs. We select a unique address space for each program. If the program starts before, we need to reload from memory. But it is first time to start, we can run it.

Before running the program, we check is it finish or not. Then we run it. If this program already finished, we need to find another running program.

For example, take a program 1's code:

```
39  ########## Prog 1 ############
40  Prog1Load:
41      li $t0, 1
42      sw $t0, RunningProcess
43
44      la $k1, Collatz
45      li $v0, 22
46      syscall
47
48      la $t1, 0x10011000
49      lw $t0, Prog1Flag
50      bnez $t0, RegLoad
51      j Prog1Run
52
53  Prog1Return:
54      lw $t0,Prog1Flag
55      addi $t0, $t0, 1
56      sw $t0, Prog1Flag
57      j Prog2Load
58
59  Prog1Run:
60      li $s4,40
61      lw $v0,Prog1Exit
62      beq $s4, $v0, ProgCheck
63
64      lw $s4, 20($t1)
65      lw $t0, 36($t1)
66      lw $t2, 44($t1)
67
68      lw $a3, ProgCounter1
69      li $v0, 19
70      syscall
71      sw $a3, ProgCounter1
72
73      sw $v0, Prog1Exit
74
75      j RegStore
```

In line 48, we save the program to memory address. In 49, 50, we the program has already started or not. If it started, we reload and go to regload. If this is first time to run, we run it.

In 60, 61, we check the program is already finish or not. If it finished, we should find the next program to run and go to progcheck. After this check, we take its program counter and run it.

After the program running, we save its return value for understand to exit or not. Then jump regstore and we save it's all registers.

PALINDROME:

We have a string array in cpp and we are using a syscall we take one string. If we take 100 string, we ask a question to user for continue. If user enter 'y', we take a new string from user, or we exit the program.

When we find palindrome, we copy the string two times. Then we go to the end of one copy. Then we start to compare char by char these strings. If there is no match, we write "it is not palindrome" and go to the beginning. If all characters match, we write "it is palindrome." Then go to the beginning.

Ahmet Yuşa Telli

151044092