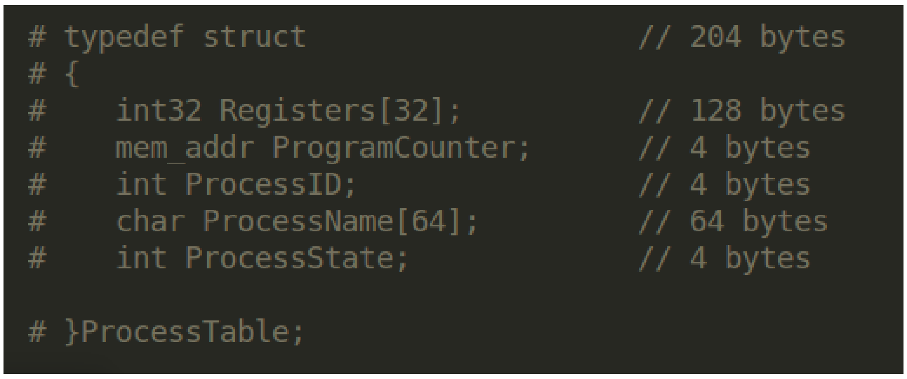
CSE 312 OPERATING SYSTEMS SPRING 2020 HOMEWORK#03

Ali Haydar KURBAN 151044058

In each micro kernel I know that how many processess will work so that making a system call by "**ALLOCATE_MEMORY_FOR_ADDRESS_OF_ASSEMBLY_STRUCT**", I can allocate some memory for assembly struct. This assembly sturct is like that:



```
# typedef struct                        // 204 bytes
# {
#     int32 Registers[32];              // 128 bytes
#     mem_addr ProgramCounter;          // 4 bytes
#     int ProcessID;                    // 4 bytes
#     char ProcessName[64];             // 64 bytes
#     int ProcessState;                 // 4 bytes

# }ProcessTable;
```

This allocated memory is for an integer array that holds the start address of the each assembly sturct. In each micro kernel I allocate memory for each process. This allocated memory could not be one after another. So that I have to keep the start address of each assembly struct. **R[REG_A3]** registers which is **$a3** was used for this syscall that holds the number of process and **18** is the syscall.

In each micro kernel I know some information about the tester processes. Such as process name, process id, process state. The unkown informations are registers and program counter. So, with **SBRK_SYSCALL** that is **9**, I allocated memory **204 byte** that is the size of the ProcessTable struct. Then I add the program into the memory (it explained in homework dpf). This addition is like that, for each micro kernel the first process is "**init**". Then I used the allocated memory that was 204 byte and I loaded the information that I know. Registers are not known so that the first **128 byte** were loaded empty, and also program counter is not kown but initially I load **0**. Then I have 204 byte memory and there are some information about the process that are loaded the memory.

Finally the assembly struct is available then I can fork. Fork is done by "**TAKE_ASSEMBLY_STRUCT_AND_CALL_FORK**" and its syscall value is **19**. **R[REG_A0]** which is **$a0** is used to take the start address of the curret assembly sturct that is allocated. With this syscall, I stored the start addres of the assembly struct in the array that was created with the "**ALLOCATE_MEMORY_FOR_ADDRESS_OF_ASSEMBLY_STRUCT**" syscall. And also I found the unkown program counter value for each process and I changed the value of it on the memory address. Then this syscall was called for each program. So that new processes are created like that.

After all programs are loaded into the memory, I made the "init" wait until all poceses terminate their jobs. Then I made the "**EXECVE**" syscall with the value of **20**. This syscall makes the first execution. It means that this syscall change the code image with the next running process instead of "init". Changing code image is like that: In the assembly sturct I load the kown informaition in the memort for each process. Then I stored information about the "init" and change the real system value such as PC and R with the next run process.

There is a **Round Robin Scheduling** mechanism when a process is interrupted by timer or it finishes its jobs. It means that with an infitine loop, look all processes that are loaded in to the memory in each micro kernel with circular way. (When a process is the last item in the loaded assembly sturcts, the next item of it, is the first item.) When it finds a ready or not started process, it puts its registers and program counter into the allocated memory for its assembly sturct and run the founded process. This run operation is like that: When I found the ready or not started process I can know the information about the its assembly struct. Such as its name, its id and so on… Then I change the real system value such as PC and R with the next run process.

"**TERMINATE_AND_EXECUTE_NEW_PROCESS**" syscall with the value of **22** is called when a program finished it jobs. (**BinarySearch.s**, **LinearSearch.s**, **Collatz.s**, **Palindrome.s**) With this this system call I made a **Round Robin Scheduling** mechanism and run the founded process that was is the ready or not_start_yet state.

"**RANDOM_INT_GENERATOR**" syscall with the value of **42** is used to generate a random number for **SPIMOS_GTU_2.s** and **SPIMOS_GTU_3.s**. It it takes an upper bound with **R[REG_A0]** which is **$a0** to generate random number and assigns the random number to **R[REG_RES]** which is **$v0**.

Each micro kernel ends with “**PROCESS_EXIT**” syscall. It deallocates memory (integer addres array) that was allocated for the start address of the each assembly structs. And prints some information about the processes and made POSIX PROCESS EXIT syscall. (exit(1))

IMPORTANT POINTS

If you want to run **LinearSearch.s**, **BinarySearch.s**, **Collatz.s** and **Palindrome.s** seperatly on the spim, you have to change “**li $v0, 22**” syscall with “**li $v0, 10**” in these files. The reason why I made it different syscall is I do not want to change syscall of spim.