Operating Systems
Constructor University
Dr. Jürgen Schönwälder

Module: CO-562
Date: 2023-09-15
Due: 2023-09-22

Problem Sheet #2

Problem 2.1: *riscv stack frame* (2+2 = 4 points)

The following C source code has been compiled by `gcc` (without optimizations).

```
1  int main(int argc, char *argv[])
2  {
3      int rc = 0;
4      return rc;
5  }
```

The generated RISC-V assembly code looks like this:

```
1  main:
2          addi    sp,sp,-48
3          sd      s0,40(sp)
4          addi    s0,sp,48
5          mv      a5,a0
6          sd      a1,-48(s0)
7          sw      a5,-36(s0)
8          sw      zero,-20(s0)
9          lw      a5,-20(s0)
10         mv      a0,a5
11         ld      s0,40(sp)
12         addi    sp,sp,48
13         jr      ra
```

a) Lookup the RISC-V instruction set and comment each instruction of the assembly code.

b) Draw a figure showing the layout of the stack frame. Show where the variables are stored and where the registers `sp` (stack pointer) and `s0` (stack frame pointer) point to during the function execution.

Problem 2.2: *bench - measure command execution time* (6 points)

Write a C program called `bench` that executes a command repeatedly in order to measure its execution time. Your implementation of `bench` should implement a command line option `-d` to set the number of seconds that `bench` should executed the command in a loop. This duration defaults to 5 seconds. The option `-w` can be used to set the number of warmup runs. The default number of warmup runs is 0. Your `bench` program should produce summary information indicating the time of the shortest run, the time of the longest run, the average runtime, the total time that has elapsed and the total number of command execution and the number of failed command executions.

Your program must use the `fork()`, `execvp()`, and `waitpid()` system calls. You are not allowed to use the `system()` library call. Here are two example executions:

```
$ ./bench -w 2 -d 4 sleep 1
Min:   1.003625 seconds         Warumps: 2
Avg:   1.007656 seconds         Runs: 4
Max:   1.013614 seconds         Fails: 0
Total: 4.030625 seconds
```

```
$ ./bench -w 2 -d 1 -- sh -c 'date > /dev/null'`
Min:   0.006297 seconds         Warumps: 2
Avg:   0.006403 seconds         Runs: 157
Max:   0.008052 seconds         Fails: 0
Total: 1.005335 seconds
```

Make sure your program properly handles all possible runtime errors and that it returns an error status to its parent process (usually the shell) in case a runtime error occured.

Use the `getopt()` function of the C library for command line option parsing. To obtain clock information, use the `clock_gettime()` function defined in `time.h` with the `CLOCK_MONOTONIC` clock identifier.