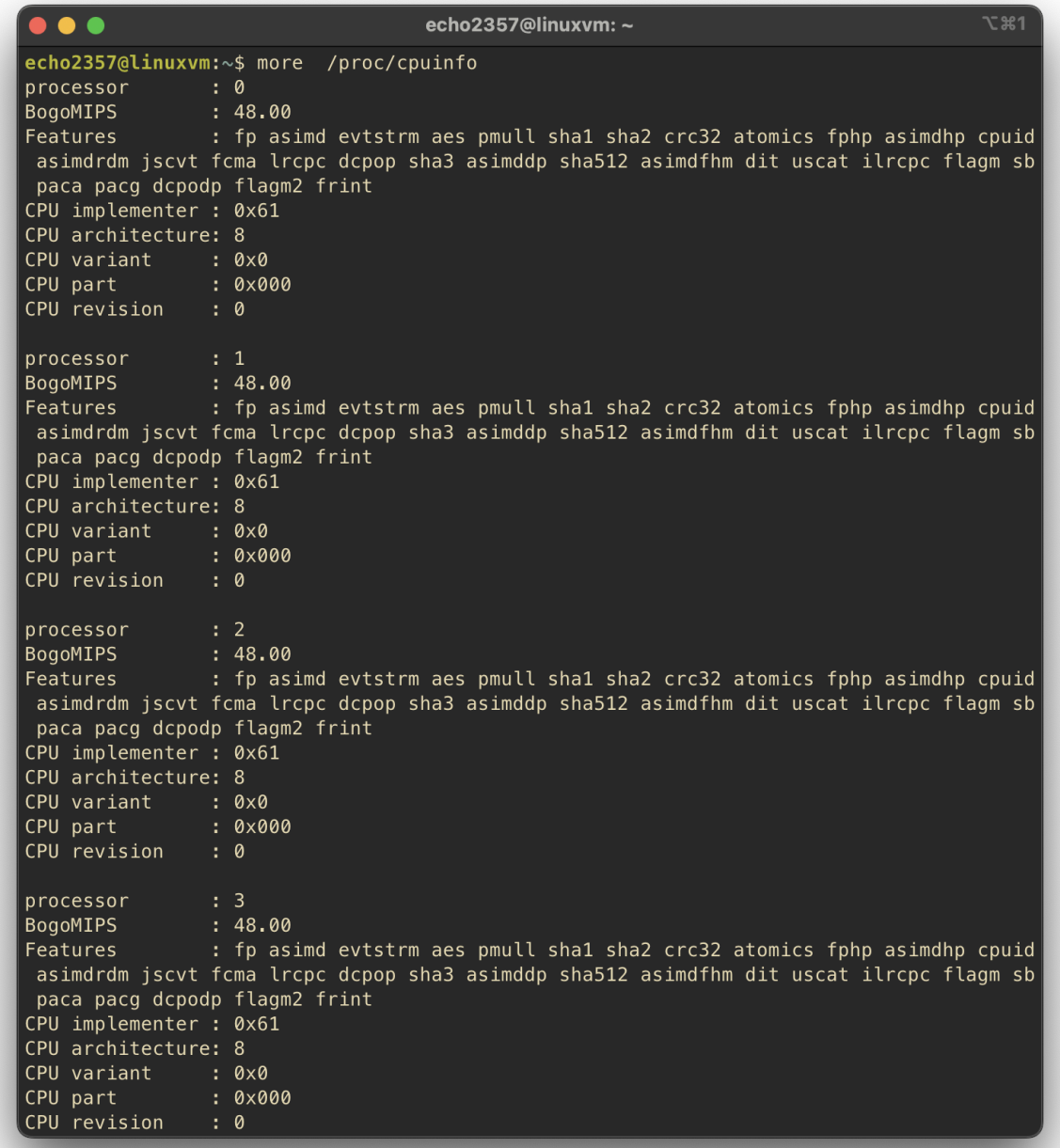
Lab 01

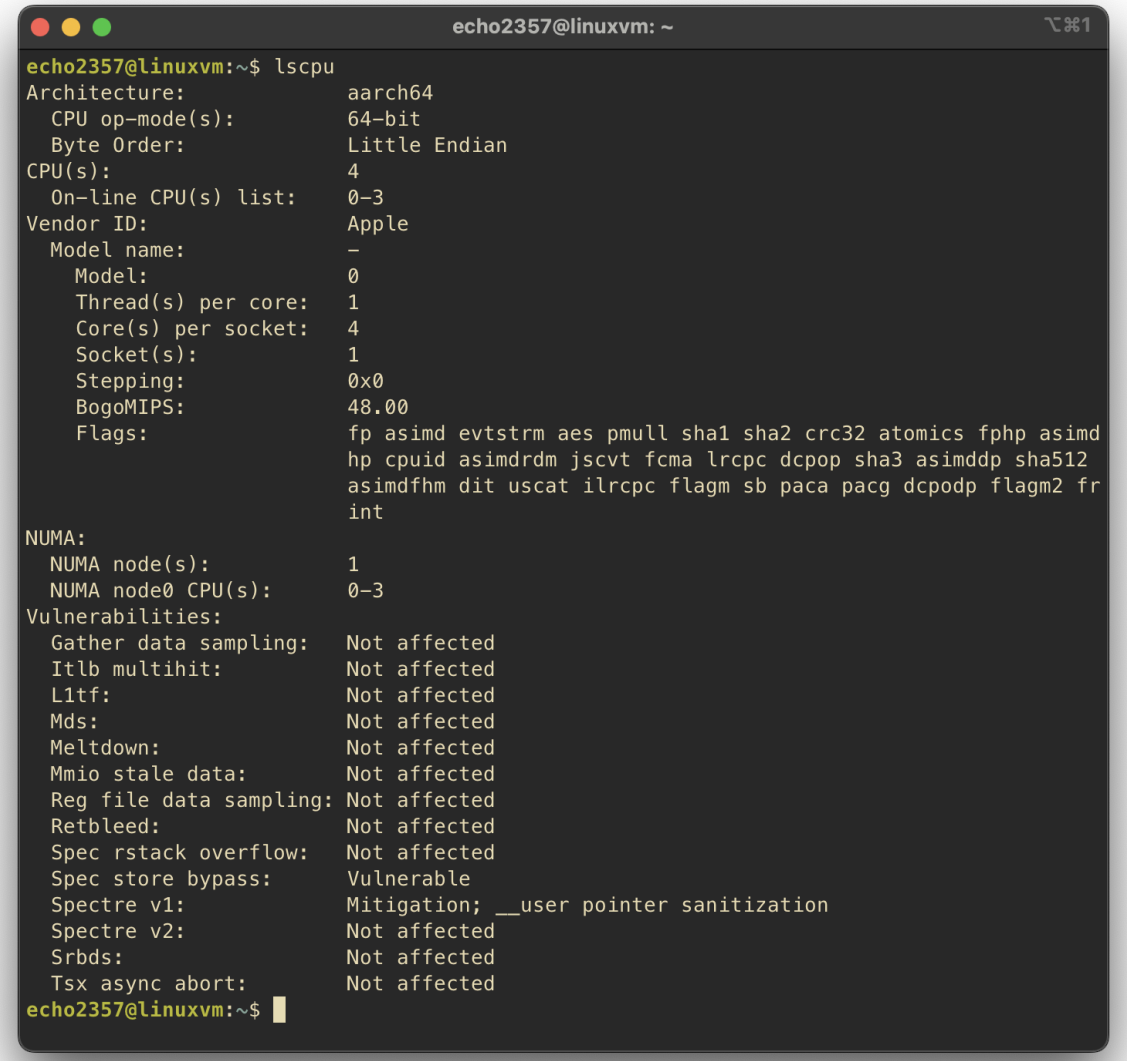
Echo Wang A01347203

Question 1

`more  /proc/cpuinfo`



`lscpu`



(a) Definitions

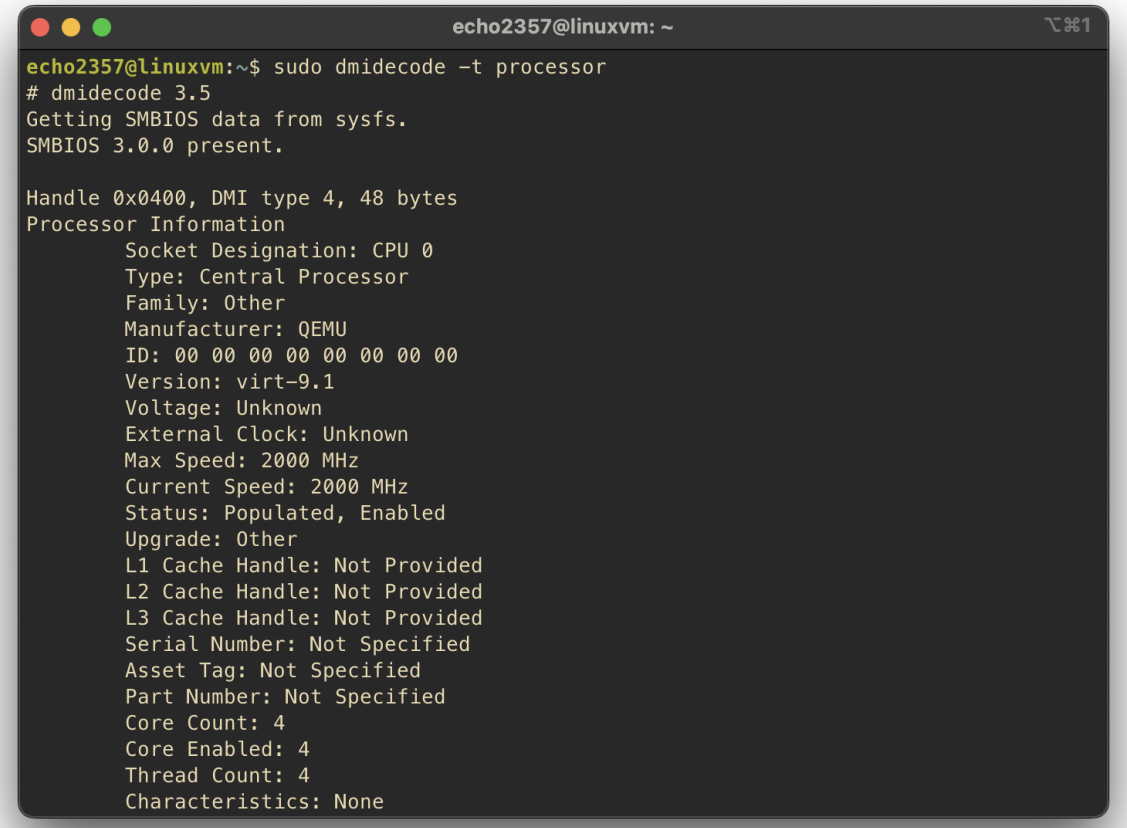
- Processor: A logical CPU shown by /proc/cpuinfo. Each processor corresponds to a thread that the OS can assign tasks to
- Core: A physical CPU unit inside the processor.
- Hyperthreading: The number of logical processors could be more than the number or physical cores, as each core can handle multiple threads. However, in the case of my VM, they seem to be the same, i.e., no hyperthreading

(b) 4

(c) 4

(d) The QEMU hypervisor seems to abstract away actual hardware details about the processor, so there's no frequency info in the output of the 2 commands mentioned above. I used " sudo dmidecode -t processor" to read DMI data

about the processor, and got this:



QEMU reports to the guest OS that the static frequency of the physical processor is 2000MHZ

(e) aarch64

(f) 3996276 kB

`cat /proc/meminfo | grep MemTotal`

```
echo2357@linuxvm:~$ cat /proc/meminfo | grep MemTotal
MemTotal:        3996276 kB
```

(g) 3032132 kB

`cat /proc/meminfo | grep MemFree`

```
echo2357@linuxvm:~$ cat /proc/meminfo | grep MemFree
MemFree:         3032132 kB
```

(h) 5894

`cat /proc/stat | grep "processes"`

```
echo2357@linuxvm:~$ cat /proc/stat | grep "processes"
processes 5894
```

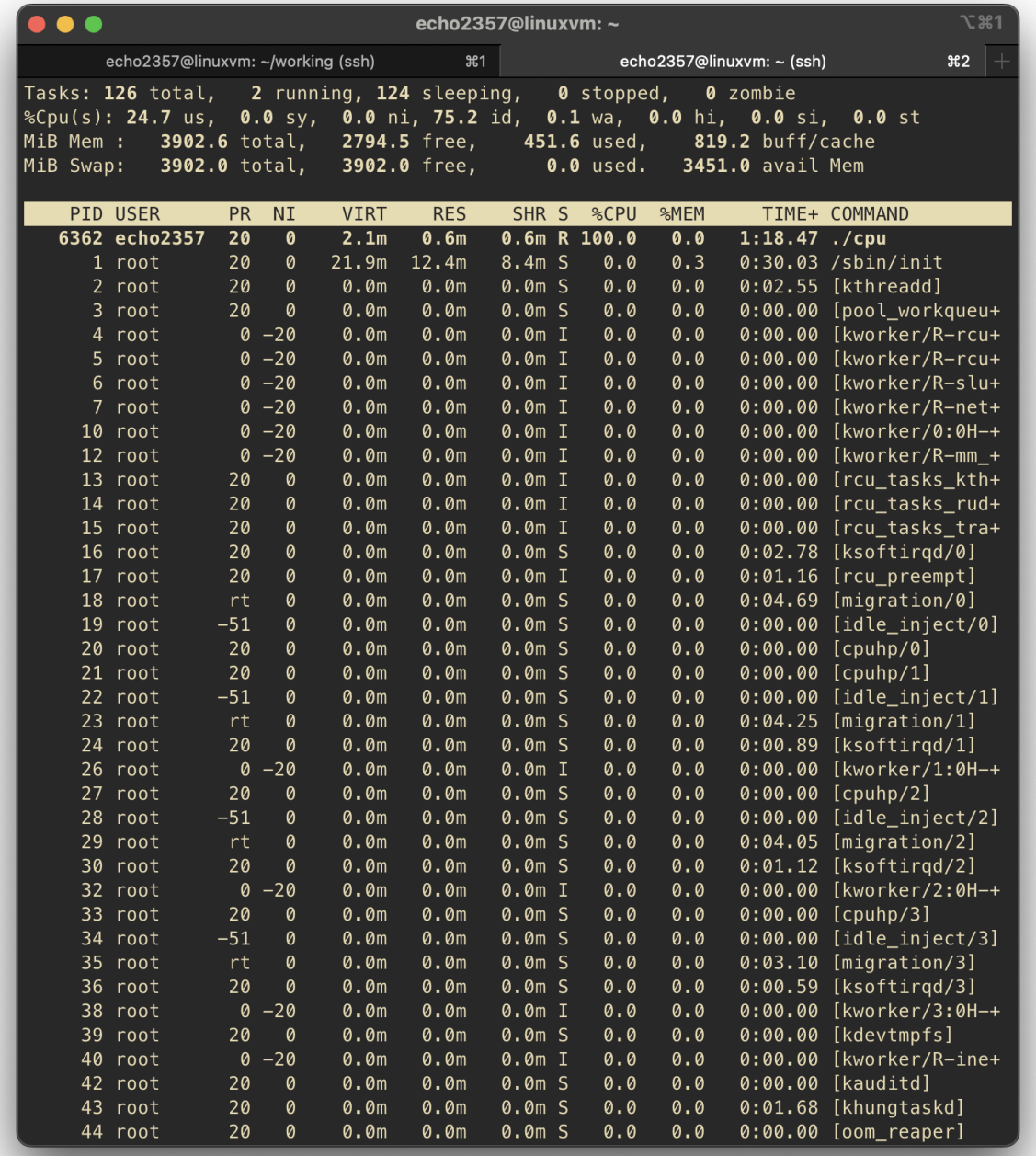
(i) 7883 voluntary context switches, 1016 nonvoluntary, 8899 in total

```
cat /proc/1/status | grep "ctxt_switches"
```

```
echo2357@linuxvm:~$ cat /proc/1/status | grep "ctxt_switches"
voluntary_ctxt_switches:        7883
nonvoluntary_ctxt_switches:     1016
```

Question 2

(a) 6342



(b) 100% CPU and 0% memory

(c) It's running, as indicated by the R in the S (status) column, which stands for "Running"

Question 3

(a)

PID of cpu-print process is 6452

`ps aux | grep cpu-print`

```
echo2357@linuxvm:~$ ps aux | grep cpu-print
echo2357    6452  0.0  0.0   2252  1280 pts/0    S+   08:03   0:00 ./cpu-print
echo2357    6529  0.0  0.0   6272  2048 pts/1    S+   08:05   0:00 grep --color=auto cpu-print
```

(b)

`ps -o ppid = -p [current PID]`

PID of 5 generations:

5645, 5644,5588, 5433, 1(init)

```
echo2357@linuxvm: ~                                    ⌥⌘1
echo2357@linuxvm: ~/working (ssh) ⌘1   echo2357@linuxvm: ~ (ssh) ⌘2
echo2357@linuxvm:~$ ps -o ppid= -p 6452
   5645
echo2357@linuxvm:~$ ps -o ppid= -p 5645
   5644
echo2357@linuxvm:~$ ps -o ppid= -p 5644
   5588
echo2357@linuxvm:~$ ps -o ppid= -p 5588
   5433
echo2357@linuxvm:~$ ps -o ppid= -p 5433
      1
echo2357@linuxvm:~$ ps -p 1
    PID TTY          TIME CMD
      1 ?        00:00:30 systemd
echo2357@linuxvm:~$
```

(c)

`ls -l /proc/6554/fd/`

```
echo2357@linuxvm:~/working$ ./cpu-print > /tmp/tmp.txt &
[1] 6554
```

```
echo2357@linuxvm:~$ ls -l /proc/6554/fd/
total 0
lrwx------ 1 echo2357 echo2357 64 Jan 10 08:15 0 -> /dev/pts/0
l-wx------ 1 echo2357 echo2357 64 Jan 10 08:15 1 -> /tmp/tmp.txt
lrwx------ 1 echo2357 echo2357 64 Jan 10 08:15 2 -> /dev/pts/0
```

FD 0: Points to the terminal normally. We didn't redirect input, so it stays the terminal

FD 1: Points to the terminal normally, but as we redirected the output with the ">" operator to a text file, now it points to the file.

FD 2: Points to the terminal by default. Since there's no redirection, it remains pointing to the terminal.

So in IO redirection, the shell modifies the file descriptors of the process to the destination specified by the commands.

(d)

```
echo2357@linuxvm:~/working$ ./cpu-print | grep hello &
[4] 6574
```

```
ps aux | grep cpu-print
ps aux | grep grep
ls -l /proc/6573/fd/
ls -l /proc/6574/fd/
```

The pipe [45823] is modified by the shell to be the output (FD 1) of the process before the pipe operator, and the input (FD 0) of the process after the pipe operator, i.e., the pipe connects 2 processes. In this case, the data produced by `cpu-print` flows into the pipe, and the `grep hello` reads from the pipe to filter the lines containing the word "hello". The pipe is actually "redirecting" the output of one process as the input of another.

(e)

Implemented by shell code: cd, history

Executables: ls, ps

Commands implemented by shell code are also called "built-in shell commands", which are usually simpler, involving basic shell operations, and faster

Commands that invoke executables are usually more complex, and also tends to take longer.