THEORY OF INFORMATION, ARCHITECTURE OF COMPUTERS AND OPERATING SYSTEMS (TIACOS)
Bioinformatics, ESCI
DAC, UPC

Final exam **Spring 2023**

There are six questions. Answer the questions in a text file. Mark clearly which question you are answering. Paper solutions are not allowed.

Grades will be published at aula.esci on **Monday, 3 July at 8.00 am.**

Review for the exam will be held in building C6, room 216, Campus Nord, UPC on **Monday, 3 July, 11.00 - 12.00 am.**

Questions

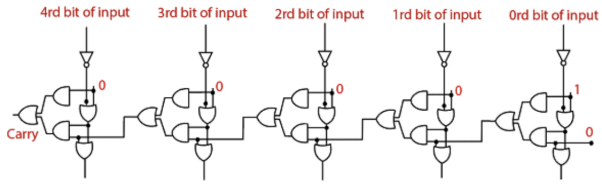
1. (1 point) **Computer architechture**
   (a) What's the first phase of CPU's operation?

   The Fetch phase

   (b) What happens during the first phase of CPU's operation?

   This is where we retrieve the instruction from memory pointed by th Instruction Pointer register (also named PC, or IAR register) and this value is copied to the Instruction Register

2. (1 point) **Integer representation**
   This is the logic circuit for finding 2's complement of the 5-bit binary number.



   (a) If the input is 10000_2, what's the output in binary?

   For 5 bits, this is the most negative number, so it is a special case. The two's complement of the most negative number representable is itself, 10000_2. As explained in classroom (slide 13, Unit Bits), you have to reverse the bits and add 1, ignoring the overflow.

   (b) What's the decimal value, in 5 bits 2's complement representation, of the integer 10000_2?

$$y = -y_{n-1} \cdot 2^{n-1} \sum_{i=0}^{n-2} y_i \cdot 2^i = -1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = -16$$

3. (0.5 points) **Operating System**
   (a) What is the goal of a cpu scheduler algorithm?

A scheduler may aim at one or more goals, for example:

- maximizing throughput (the total amount of work completed per time unit);
- maximizing fairness (equal CPU time to each process, or more generally appropriate times according to the priority and workload of each process).

(b) In this context, what's the meaning of preemption?

In cpu scheduling, preemption is the act of temporarily interrupting an executing task, with the intention of resuming it at a later time.

4. (0.5 points) **Memory**

(a) What is thrashing?

Thrashing occurs when virtual memory management is in a constant state of paging and page faults.

(b) What would you do to solve it?

Increasing RAM or killing processes.

5. (3 points) **Processes**.

Read the following Python code carefully and answer the questions. Let's assume there are no exceptions during the execution of this code.

```
# global variables
count = 3
q = ["NOT PASSED","PASSED"]
i = 1
examen = q[0]
# handler function for signal USR1
def handler (signum, frame):
    global i
    i = 2
    return 0
# child process work
def work():
    global examen
    examen = q[1]
    os.execlp("grep","grep","wilkes","/etc/passwd")
    examen = q[0]
    return 0

signal.signal(signal.SIGUSR1,handler)
while count > 0:
    pid = os.fork()
    if pid == 0:
        os.kill(os.getpid(),signal.SIGUSR1)
        examen = q[work()]
    else:
        r,s = os.wait()
        examen = q[i-(s>>8)]
    count -= 1
    print(os.getpid(),"-",examen)
```

(a) (0.5 points) How many processes are created?

3 children + parent

(b) (0.5 points) What happens when the parent process receives the signal SIGCHLD?

Nothing, by default it is ignored and in this code it is not reprogrammed.

(c) (0.5 points) These processes are executed concurrently or sequentially?

Sequentially. Parent process waits for the child exits before create another one.

(d) (0.5 points) In any child process, what's the last value of the variable examen?

`examen = q[1]` (line 19) after that the system call `exec` changes the code and examen disappears

(e) (1 point) Change the argument of just one system call in order to modify the printed message from NOT PASSED to PASSED. Which line has you changed?

`PASSED` string is in position 1 of the `q` array. Parent process prints `q[i-(s<<8)]`, therefore, in other to change the output we can change `i` or `s`. So, there are two solutions to this question, change `s` or `i`, but changing only one argument of a system call.

- Change `i`: In line 23, the child process is sending a SIGUSR1 signal to itself, however the child process does not print anything. If it sends a SIGUSR1 signal to its parent, the handler function will be executed and the global variable `i` will be 2. So, changing `getpid()` by `getppid()` is an answer.
- Change `s`: And this solution has two folds:
  - Change `signal.SIGUSR1`: Also in line 23, if you send any signal that provoques the termination of the child process, the value of `s` in the parent process after `wait()` will change.
  - In line 15. That's the return of the execution of `grep`. As the man page says, `grep` returns 0 if it finds something and returns 1 if it doesn't. In this case, is returning 1, because user "wilkes" does not exit. In order to return 0, we have to ask `grep` for an existing user. For instance, "root".

6. (4 points) **Files**.

We have a text (ascii) file, called bases.txt, containing strings of bases separated by a blank space. It looks like this:

```
CTGA AGTC CCTA AAGG
```

We want to program a function in Python, called `read_word` which, given a valid file descriptor, returns the next string of bases. We have three different versions of the function `read_word`.

Carefully analyse the code and give justified answers to the questions.

Listing 1: version A

```
def read_word(fd):
    word=b''
    found=False
    c=os.read(fd,2)
    while (len(c)>1) and not found:
        if (c[1]==32):
            found=True
        else:
            word+=c
            c=os.read(fd,2)
    return word
```

Listing 2: version B

```
def read_word(fd):
  word=b''
  c=os.read(fd,2)
  while (len(c)>1):
      if (c[1]==32):
          break
      word+=c
      c=os.read(fd,2)
  return word
```

Listing 3: version C

```
def read_word(fd):
    word=b''
    found=False
    c=os.read(fd,2)
    while (len(c)>1) and not found:
        word+=c[0].to_bytes(1,"little")
        os.lseek(fd,-1,os.SEEK_CUR)
        if (c[0]==32):
            found=True
        else:
            c=os.read(fd,2)
    return word
```

(a) (2 points) Which version(s) of `read_word` works correctly?

> Only version C works properly, because the three versions read 2 bytes each loop, but only C checks byte by byte.

(b) (2 points) Assuming you have a version of `read_word` that correctly returns a whitespace-terminated base string, program a Python script that opens the bases.txt file for reading and, using only the `read_word` function and any system calls, reads all base strings and writes them to standard output.

```
fd = os.open("bases.txt", os.O_RDONLY)
word = read_word(fd)
while (len(word)>0):
    os.write(1,word)
    word = read_word(fd)
os.close(fd)
```