I/O Systems
Solved exercises

Exercise 1. You want to develop a controller for a microwave oven. The controller features a 32-bit processor, common I/O map, and the RISC-V32 instruction set. To this processor is connected an I/O module, which controls the operation of the oven. The module has the following five 32-bit registers:

- Register with address 1000. In this register, the value corresponding to the countdown is loaded in seconds.
- Register with address 1004. In this register a 1 is loaded when you want to start the countdown. As long as there is a 0, the countdown does not start.
- Register with address 1008. When the countdown reaches 0, in this register the controller stores a 1. While the countdown is being performed, the value that is stored in this record is 0.
- Register with address 1012. In this register the value corresponding to the desired power is loaded. The value is a 32-bit integer in two complements.
- Register with address 1016. When the value 1 is loaded into this register, the microwave is put into operation with the power stored in the previous register. When a 0 is loaded in this register, the microwave stops regardless of whether the end of the countdown has been reached.

You want to write a function in assembler, called `Oven_Handler` that allows you to control the operation of the oven. This function will receive two input parameters: an integer value indicating the number of seconds that the oven must remain on and an integer value that represents the power of the oven. The function will be responsible for programming the previous controller with the power and seconds passed as parameters, taking care of starting the oven and turning it off when the necessary time has elapsed. The function will return the following possible values:

- A value equal to 0 that represents that there has been no error and that it will be returned when the function has stopped the oven.
- A value equal to -1, when the power passed as a parameter is less than 100 and greater than 1000. In this case the oven controller is not to be programmed and the function must immediately return to the value -1.

It is requested:

a) Indicate in which registers the parameters are to be passed and in which registers the result should be returned.
b) Invoke the above function for a power value of 800 W and a duration of 90 seconds. Then type the code that allows you to print the result returned by the function on the screen.
c) Type the code for the `Oven_Handler` function.

Solution:

```
Oven_Handler:        li    t0, 100
                     blt   a0, t0, end_error
                     li    t0, 1000
                     bgt   a0, t0, end_error
                     la    t2, 1000
                     sw    a1, 0(t2)          ; countdown
                     la    t2, 1012
                     sw    a0, 0(t2)          ; power
                     li    t0, 1
                     la    t2, 1016
                     sw    t0, 0(t2)          ; motor starts
                     la    t2, 1004
                     sw    t0, 0(t2)          ; start countdown
          loop:      la    t2, 1008
                     lw    t1, 0(t2)          ; read state countdown
                     beq   t1, x0, loop
                     la    t2, 1016
                     sw    $0, 0(t2)          ; motor stops
                     mv  a0, x0
                     jr  ra
      end_error:     li  a0, -1
                     jr  ra
```

Exercise 2. You want to develop a driver for a device that allows you to measure a person's blood pressure. The associated I/O module connects to a computer that has a 32-bit processor, common I/O map, and RISC-V32 instruction set. The module has the following 32-bit registers:

- Register with address 1000. When the value 1 is loaded into this register, the module is told that it wants to start a pressure measurement.
- Register with address 1004. When the measurement has finished in this register, a 1 is stored, while the device is performing the measurement, its value is 0.
- Register with address 1008. When the device finishes the pressure measurement, the value corresponding to the systolic pressure is stored in this register.
- Register with address 1012. When the device finishes the pressure measurement, the value corresponding to the diastolic pressure is stored in this register.

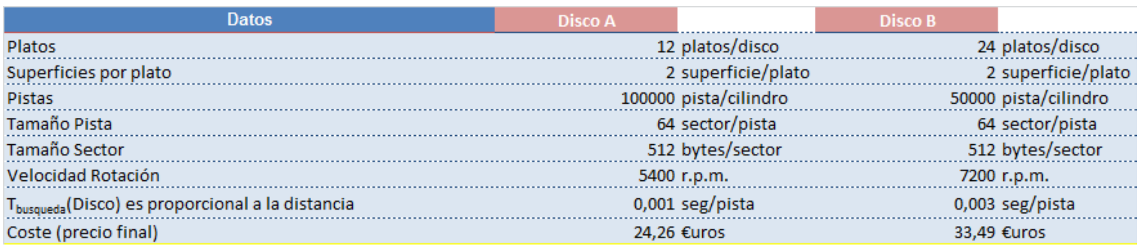
Design and implement an assembler function, called `Preassure_handler` that allows you to control the operation of this computer. This function will not receive any input parameters. The function will be responsible for measuring the pressure and will return as output parameters the value of the systolic pressure and the value of the diastolic pressure.

Solution:

```
Preassure_handler:      li    t0, 1
                        la    t2, 1000
                        sw    t0, 0(t2)

            loop:
                        la    t2, 1004
                        lw    t0, 0(t2)
                        beq   x0, t0, loop

                        la    t2, 1008
                        lw    v0, 0(t2)
                        la    t2, 1012
                        lw    v1, 0(t2)

                        jr    ra
```

Exercise 3. You want to buy a hard drive for a home computer that gives the best quality/price ratio. Since the disc will be used intensively for recording and playing movies, the average access time and storage capacity is considered more important than the cost per MB of the disc.

Two possible candidate disks are available with the following parameters:



Datos	Disco A		Disco B	
Platos	12	platos/disco	24	platos/disco
Superficies por plato	2	superficie/plato	2	superficie/plato
Pistas	100000	pista/cilindro	50000	pista/cilindro
Tamaño Pista	64	sector/pista	64	sector/pista
Tamaño Sector	512	bytes/sector	512	bytes/sector
Velocidad Rotación	5400	r.p.m.	7200	r.p.m.
$T_{busqueda}$(Disco) es proporcional a la distancia	0,001	seg/pista	0,003	seg/pista
Coste (precio final)	24,26	€uros	33,49	€uros

Disk A is equipped with an older technology than disk B, as can be seen by the rotation speed of the disk.
It is requested:

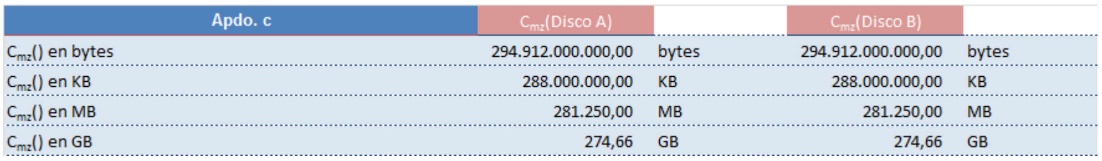
a) Calculate the capacity of each disk if it is formatted with four zones of equal size with densities of 64, 128, 256 and 512 sectors per track.
b) Calculate the average access time of each disk with this format by zones.
c) Calculate the cost per MB of each disk and justify, based on all the calculated data, which is the best disk to buy.

Solution:

a) Calculate the capacity of each disk if it is formatted with four zones of equal size with densities of 64, 128, 256 and 512 sectors per track.

Keep in mind that the formula to use is:

$$\#\text{ platters} * \#\text{ surfaces} * \left(\frac{\#\text{ tracks}}{4}\right) * (64 * \text{Size(sector)} + 128 * \text{Size(sector)} + 256 * \text{Size(sector)} + 512 * \text{Size(sector)})$$



Apdo. c	C_{mz}(Disco A)		C_{mz}(Disco B)	
C_{mz}() en bytes	294.912.000.000,00	bytes	294.912.000.000,00	bytes
C_{mz}() en KB	288.000.000,00	KB	288.000.000,00	KB
C_{mz}() en MB	281.250,00	MB	281.250,00	MB
C_{mz}() en GB	274,66	GB	274,66	GB

b) Calculate the average access time of each disk with this format by zones.

Apdo. b: Cálculos previos		
$T_{busqueda}$()	$T_{busqueda}$(Disco A)	$T_{busqueda}$(Disco B)
Peor Caso: de la pista 0 a la pista 99999	100000 pistas	5000 pistas
Mejor Caso: se está en la pista	0 pistas	0 pistas
Caso medio: se está a la mitad de pistas entre ambas	50000 pistas	2500 pistas
$T_{busquedapromedio}$(Disco A)=Sumatorio (Caso*su tiempo)/3casos	50 segundos	7,5 segundos
$T_{busquedapromedio}$(Disco A)=Sumatorio (Caso*su tiempo)/3casos	50000 mseg.	7500 mseg.

$T_{rotacion}$()	$T_{rotacion}$(Disco A)	$T_{rotacion}$(Disco B)
Velocidad rotación en segundos(v_{rot})	90 r.p.seg.	120 r.p.seg.
Tiempo una rotación ($1/v_{rot}$) en segundos	0,011111111 segundos	0,008333333 segundos
Tiempo una rotación ($1/v_{rot}$) en mseg	11,11111111 mseg.	8,333333333 mseg.

Calculando el $T_{transferencia}$		
$T_{transferencia}$() con zonas de igual número de pistas y diferente densidad: 64, 128, 256 y 512 sectores/pista		
$T_{transferencia}$() con zona de **64 sectores/pista**	Disco A	Disco B
Tiempo recorrer todos los sectores de una pista	11,11111111 mseg.	8,333333333 mseg.
Número de sectores por pista	64 sector/pista	64 sector/pista
$T_{transferencia}$() promedio con zona de **64 sectores/pista**	0,173611111 mseg.	0,130208333 mseg.

$T_{transferencia}$() con zona de **128 sectores/pista**	Disco A	Disco B
Tiempo recorrer todos los sectores de una pista	11,11111111 mseg.	8,333333333 mseg.
Número de sectores por pista	128 sector/pista	128 sector/pista
$T_{transferencia}$() promedio con zona de **128 sectores/pista**	0,086805556 mseg.	0,065104167 mseg.

$T_{transferencia}$() con zona de **256 sectores/pista**	Disco A	Disco B
Tiempo recorrer todos los sectores de una pista	11,11111111 mseg.	8,333333333 mseg.
Número de sectores por pista	256 sector/pista	256 sector/pista
$T_{transferencia}$() promedio con zona de **256 sectores/pista**	0,043402778 mseg.	0,032552083 mseg.

$T_{transferencia}$() con zona de **512 sectores/pista**	Disco A	Disco B
Tiempo recorrer todos los sectores de una pista	11,11111111 mseg.	8,333333333 mseg.
Número de sectores por pista	512 sector/pista	512 sector/pista
$T_{transferencia}$() promedio con zona de **512 sectores/pista**	0,021701389 mseg.	0,016276042 mseg.

	$T_{transferencia}$(Disco A)	$T_{transferencia}$(Disco B)
$T_{transferencia}$() con zonas (promediando)	0,081380208 mseg.	0,061035156 mseg.

Calculando T_{acceso} requerido en los apartados b y d	Disco A	Disco B
$T_{busqueda}$()	50000 mseg.	7500 mseg.
$T_{rotacion}$()	11,11111111 mseg.	8,333333333 mseg.
$T_{transferencia}$() con zonas (promediando)	0,081380208 mseg.	0,061035156 mseg.
T_{acceso}() con Zonas	50011,19249 mseg.	7508,394368 mseg.

c) Calculate the cost per MB of each disk and justify, based on all the calculated data, which is the best disk to buy.

The cost per GB of each disk is:

Calculando el coste por almacenamiento de cada disco	Disco A		Disco B	
Tamaño con CAV	73,24	GB	73,24	GB
Tamaño con pistas de tamaño variable	274,66	GB	274,66	GB
Coste por GB con CAV	0,33	€uros/GB	0,46	€uros/GB
Coste por GB con pistas de tamaño variable	0,09	€uros/GB	0,12	€uros/GB

Since both disks would have the same size, either as CAV or with variable distribution of sectors per track, the size of the disk is not a significant data for the decision.

Noting that the access time of disk A is greater than the access time of disk B but the cost per GB is lower on disk A, the decision will have to be made between which feature is more desirable: if it is the speed then you will have to acquire disk B, if it is the economy then you will have to acquire disk A.

Exercise 4. Be a 32-bit 500 MIPS computer, with the RISC-V32 instruction set and a common input/output map, with a clock frequency of 1 GHz. This computer has a device that produces 5000 interruptions per second. Each interruption involves the execution of an interruption handler routine that requires 2000 machine instructions. Calculate the percentage of time this computer spends dealing with interruptions from this device.

Solution:

The device interrupts 5000 times per second, therefore the number of instructions that are executed per second as part of the treatment of interruptions is 2000 x 5000 = 10 MIPS. The percentage of time spent attending to interruptions is 10/500 x 100 = 2%.

Exercise 5. You want to develop a controller for a traffic light. The controller features a 32-bit CPU, separate I/O map, and RISC-V32 instruction set. Two I/O modules are connected to this CPU. The first is a stopwatch and the second is the I/O module that controls the operation of the traffic light. The stopwatch module has the following three registers:

- Register with address 1000. In this register, the value corresponding to the countdown is loaded in seconds.
- Register with address 1004. In this register a 1 is loaded when you want to start the countdown.
- Register with address 1008. When the countdown reaches 0, a 1 is charged in this register. While the countdown is being performed the value of this register is 0.

The I/O module that controls the traffic light has three registers:

- Register with address 1012. This register encodes the value corresponding to the color of the traffic light: 100 for red, 010 for yellow and 001 for green.
- Register with address 1016. It is a register that indicates the duration in seconds corresponding to red.
- Register with address 1020. It is a register that indicates the duration corresponding to yellow, which occurs in the red-green transition.
- Register with address 1024. It is a register that indicates the duration in seconds of the green.

It is requested:

a) Type the assembler program that controls the operation of this semaphore. The traffic light always starts its operation in red.
b) What inefficiency is identified in the previous program? How could it be resolved?

Solution:

```
RED:
      li    t0, 100
      OUT   t0, 1012
      IN    t0, 1016
      OUT   t0, 1000
      IN    t0, 1
      OUT   t0, 1004
RED_WAIT:
      IN    t0, 1008
      li    t1, 1
      beq   t0, t1, RED_WAIT
YELLOW:
      li    t0, 010
      OUT   t0, 1012
      IN    t0, 1020
      OUT   t0, 1000
      IN    t0, 1
      OUT   t0, 1004
YELLOW_WAIT:
      IN    t0, 1008
      li    t1, 1
      beq   t0, t1, YELLOW_WAIT
GREEN:
      li    t0, 010
      OUT   t0, 1012
      IN    t0, 1024
      OUT   t0, 1000
      IN    t0, 1
      OUT   t0, 1004
GREEN_WAIT:
      IN    t0, 1008
      li    t1, 1
      beq   t0, t1, GREEN_WAIT
      beq   x0, x0, RED
```

The main cause of inefficiency is CPU consumption in the meter timeout. The solution would be the use of I/O through interruptions.

Exercise 6. A computer with a clock frequency of 3 GHz has a connected hard drive with a maximum transfer rate of 20 megabits/s. The average latency between sending requests to disk and starting the transfer is 10 milliseconds. Transfers are made by programmed input/output. Preparing and submitting a reading order requires 3000 clock cycles. Read requests are for 4 KB blocks. You want to know the total execution time of an input/output operation.

Solution:

The total Tt execution time of an input/output operation will be given by the Tp preparation time, the latency until the start of the Tl transfer and the Ttr transfer time.

Tp = 3000 cycles / $3*10^9$ cycles/s = 10^{-6} secs.
Tl = $10*10^{-3}$ secs.
Ttr = (4*1024*8) bits /$20*10^6$ bits/sec = 32768 / $2*10^7$ = 0.0016384 secs.

Exercise 7. Be a 32-bit computer similar to the RISC-V32 used in the laboratory of the computer structure course. This computer has a common memory map for main memory and input/output. The memory address space is in the 0x00000000 – 0x7FFFFFFF range. The address space for input/output is in the 0x80000000 – 0xFFFFFFFF range.

A supermarket cash register has been connected to this computer that has a drawer where the money is stored, a printer and a keyboard. A barcode reader has also been connected.

The cash register output module has the following registers:

- WAIT_KEYBOARD (0x90000000 address): read-only register in which you can check the status of the keyboard, which can be WAITING_KEY (value 0) PRESSED_KEY (value 1).
- DATA_KEYBOARD (address 0x90000004): register in which the code of the last key pressed is stored. Possible values are INIT_KEY (1) and TOTAL_KEY (2).
- STATUS_DRAWER (address 0x90000008): read-only register in which you can check the status of the drawer, which can be CLOSED (value 0) or OPEN (value 1).
- PRINTER_STATUS (address 0x9000000C): read-only register in which you can check the status of the printer that can be WAITING (value 0) or PRINTING (value 1).
- PRINTER_CONTROL (0x90000010 address): register in which orders can be written WAIT_KEY (0), OPEN_DRAWER (1) or PRINT (2).
- DATA_PRINTWORK (addresses 0x90002000 and following): set of 14 registers where the ASCII codes of the barcode to be printed (first 13 records) and the price (fourteenth IEEE 754 record of simple precision) will be stored.

When you want to read a key, the module is supplied with the command WAIT_KEY and the keyboard status becomes WAITING_KEY, until a key is pressed (PRESSED KEY state) and its code is stored in the DATA_KEYBOARD register. Once the key code has been consulted, it will be necessary to send a new command WAIT_KEY to start reading a new key. If at any time you want to open the money drawer, you must send the order OPEN_DRAWER so the drawer will go from CLOSED to OPEN. When the drawer is manually closed, its status will return directly to CLOSED. Finally, to print a barcode and its price, these data will be placed in the DATA_PRINTWORK register and the PRINT order will be given so that PRINTER_STATUS goes from WAITING to PRINTING. When the line has been printed it will return to the WAITING state. If you want to print a total, the barcode will be indicated with the 13 digits with ASCII code equal to 0.

The barcode reader input/output module has the following registers:

- STATUS (0x8000100A address): read-only register in which the status of the device can be checked, which can be DISABLED (value 0), WAITING_READ (value 1), DONE_READ (value 2).
- CONTROL (0x8000100B address): register in which you can write the commands DEACTIVATE (value 0), ACTIVATE (value 1) or READ (value 2).
- DATA (addresses 0x8000100C and following): set of 13 registers containing the ASCII codes of the 13 digits of the last barcode read.

Initially the device is OFF, when the order ACTIVATE is supplied, the device goes to the WAITING_READ state. After performing an optical reading of a barcode, the device will go to the DONE_READ state and the barcode read will be stored in the 13 data register. When you want to make a new reading it will be necessary to supply the READ command, which will return to the WAITING_READ state. When you want to finish using the device, the COMMAND DEACTIVATE will be supplied, so the device will go to the OFF state.

It is requested:

a) Write a function using the RISC-V32 assembler that starts reading barcodes when the INIT_KEY key is pressed and reads until the TOTAL_KEY key is pressed. The codes will be stored in consecutive memory positions from the memory address passed in the a0 parameter. Each barcode will begin to be stored in a word-aligned address and using consecutive bytes from there. The function will return the number of codes read from v0.

b) What input/output model do the devices presented in the statement involve?

c) Which input/output model would be most appropriate for the barcode reader? Why?

d) If the cash register used three input/output modules (one for the drawer, one for the keyboard and one for the printer) Which input/output model would you select for each?

Solution:

a)

```
READ_CODES:
   addi  sp, sp -32
   sw    s0, 0(sp)
   sw    s1, 4(sp)
   sw    s2, 8(sp)
   sw    s3, 12(sp)
   sw    s4, 16(sp)
   sw    s5, 20(sp)
   sw    s6, 24(sp)
   sw    s8, 28(sp)

   mv    s7, a0

   li    s0, WAIT_KEY
   la    t0, PRINTER_CONTROL
   sw    s1, 0(t0)
   li    s2, WAITING_KEY

WHILE_NOT_PRESSED
   la    t0, WAIT_KEYBOARD
   lw    s1, 0(t0)
   beq   s0 s1, WHILE_NOT_PRESSED
```

```
    li     s0, INIT_KEY
    la     t0, DATA_KEYBOARD
    lw     s1, 0(t0)
    bne    s0, s1, READ_CODES

    li     s0, ACTIVATE
    la     t0, BAR_CONTROL
    sw     s0, 0(t0)

    li     s0, WAITING_READ

 WHILE_NOT_ACTIVE:
    la     t0, BAR_STATUS
    lw     s1, 0(t0)
    bne    s0, s1, WHILE_NOT_ACTIVTE

    li     s0, WAIT_KEY
    la     t0, PRKINTER_CONTROL
    sw     s0, 0(t0)

    li     s0, DONE_READ
    li     s1, PRESEED_KEY

 WHILE_WAIT_EVENT
    la     t0, WAIT_KEYBOARD
    lw     s2, 0(t0)
    la     t0, BAR_STATUS
    lw     s3, 0(t0)
    beq    s3, s1, CODE_PROCESSING
    beq    s2, s1, KEY_PROCESSING
    j      WHILE_WAIT_EVENT

 CODE_PROCESSING:
    la     s4, DATA_BARS
    li     s6, s3

 BAR_LOOP:
    lb     s5, 0(s4)
    sb     s5, 0(s4)
    addi   s4, s4, 1
    addi   s7, s7, 1
    addi   s6, s6, -1
    bne    s6, x0, BAR_LOOP
    addi   s7, s7, -3
    li     s5, READ
    la     t0, BAR_CONTROL
    sw     s5, 0(t0)
    bne    s2, t0, WHILE_WAIT_EVENT

 KEY_PROCESSING:
    li     s5, TOTAL_KEY
    la     t0, WAIT_KEYBOARD
    lw     t6, 0(t0)
    beq    s5, s6, THE_END
    li     s5, WAIT_KEY
    la     t0, PRINTER_CONTROL
    sw     s5, 0(t0)
    j      WHILE_WAIT_EVENT
```

```
THE_END:
   li     s0, DEACTIVATE
   la     t0, BAR_CONTROL
   sw     s0, 0(t0)
   li     s0, DEACTIVATE
WHILE_NOT_DEACTIVZTE:
   la     t0, BAR_STATUS
   lw     s1, 0(t0)
   bne    s0, s1, WHILE_NOT_DEACTIVE
   sub    a0, s7, a0
   li     t0, 4
   div    a0, a0, t0


   lw     s0, 0(sp)
   lw     s1, 4(sp)
   lw     s2, 8(sp)
   lw     s3, 12(sp)
   lw     s4, 16(sp)
   lw     s5, 20(sp)
   lw     s6, 24(sp)
   lw     s8, 28(sp)

   addi   sp, sp 32

   jr     ra
```

b) Programmed input/output

c) Direct access to memory. It would be the one that would require the least CPU occupancy, notifying when the code was stored in main memory.

d) One option would be:

- Drawer: Input/output due to interruptions
- Keyboard: Input/output due to interruptions.
- Printer: Input/output by direct memory access.