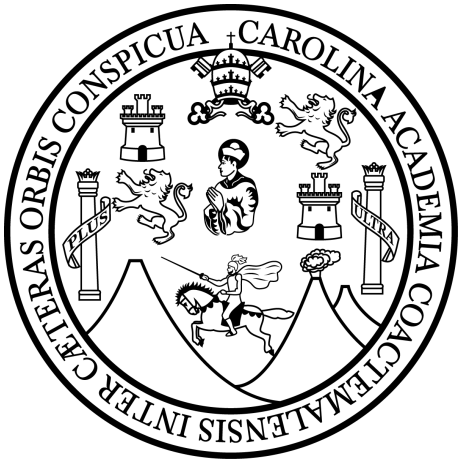
UNIVERSIDAD SAN CARLOS DE GUATEMALA

CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA CIENCIAS Y SISTEMAS



LABORATORIO DE SISTEMAS OPERATIVOS 2

"**OCTAVO SEMESTRE"**

ING.: FRANCISCO ROJAS

ESTUDIANTE: LUIS ESTUARDO BOLAÑOS GONZÁLEZ - 201731766

TRABAJO: PROYECTO I - DOCUMENTACIÓN OFICIAL

FECHA: 02 de septiembre de 2,021

DESCRIPCIÓN

El proyecto pasa por la creación de un interruptor hecho dentro de una galleta protoboard, dicho circuito conectado al computador a través de un cable usb con conexión a puerto serial, donde el la ranura usb irá conectada al computador mientras que los puntos del puerto serial hacia el protoboard, el circuito tendrá funcionamiento mediante código assembler en el que se programe la interrupción de memoria que será presentada al momento que el interruptor sea pulsado, a esto se agrega el hecho de que hay que agregar un intervalo de tiempo donde si el interruptor es pulsado dentro de ese intervalo se agregara al stack de pulsaciones y aparecerá en pantalla el tamaño del stack, si en caso el interruptor es pulsado pasado el intervalo de tiempo este contará como un nuevo stack y esa pulsación será su primer elemento. El propósito de este proyecto es el buen manejo de interrupciones de memoria, esto reflejado en las pulsaciones del interruptor, junto a esto el manejo y conocimiento del lenguaje ensamblador.

REQUERIMIENTOS TECNICOS

- Usar lenguaje ensamblador para la programación del circuito.
- Uso de sistema operativo linux para el host.
- Conexión estable entre máquina y circuito.
- Usar como puente de conexión entre máquina y circuito un cable usb a puerto serial.

HERRAMIENTAS

- Sistema operativo Linux (cualquiera).
- Lenguaje de programación Assembler.
- 1 cable usb con puerto serial.
- 1 protoboard.
- 1 capacitor de cerámica de 0.1 microFaradios.
- 1 resistencia de 10k.
- Cables jumper.
- 1 interruptor.

MARCO TEÓRICO

IRQ(INTERRUPCIONES POR HARDWARE)

¿Qué significa IRQ, qué son las interrupciones de hardware? IRQ es un acrónimo desde las palabras inglesas Interrupt Request, traducidas en castellano como solicitud de interrupción o interrupción de hardware.

Cuando un periférico, (por ejemplo una impresora) u otro dispositivo hardware (e.g. una tarjeta de sonido), necesitan "comunicarse" con la CPU utilizan una líneas de notificación preestablecidas denominadas Líneas de Interrupción (Interrupt Request Line).

Una CPU puede estar ocupada procesando billones de operaciones por segundo, lo que hace una IRQ es avisar de una nueva tarea pendiente de ser examinada. El procesador, una vez ejecutada la tarea solicitada con la IRQ, vuelve a su anterior operación. Las IRQ’s disponen de canales físicos dedicados en las placas base, cada uno con un nivel de prioridad y conectados a la CPU con pins.

Funcionamiento

¿Para qué sirven las IRQ? Los dispositivos hardware que necesitan ejecutarse transmiten una IRQ al procesador para llamar su atención. La tarjetas de red, de video, de sonido, un módem, los adaptadores SCSI, los dispositivos de tipo IDE/ADE, los periféricos USB, por puerto paralelo o serie, todos disponen de un canal prioritario para comunicarse con la CPU denominado "Número de IRQ".

Conflictos

Por definición se genera un conflicto entre IRQ cuando dos o más dispositivos de hardware tratan de utilizar la misma línea de interrupción. Esto puedo ocurrir por ejemplo al instalar un nuevo periférico o al añadir un componente hardware (conflicto hardware). Para evitar que un ordenador o un portátil incurran en error de hardware es necesario configurar correctamente las líneas de interrupción.

En los actuales PC la modificación y la configuración de IRQ es prácticamente automática, la BIOS y el SO se encargan de ello. En los años anteriores no era así: en los antiguos sistemas a 8 bits había sólo 8 direcciones IRQs disponibles y era preciso cambiar las IRQ manualmente (por medio de jumpers) como por ejemplo en las tarjetas ISA (legacy ISA). En la mayoría de sistemas a 16 bits este número subió a 15 (y no 16) añadiendo a las primeas líneas IRQ los sucesivos interrupt request 8-15, conectados a la primera serie a través del IRQ 2 que se quedaba inutilizable al ser un interrupción de "cascada" o sea puente entre 0-6 y 9-15.

Gracias a las controladoras PCI, que permitían compartir la misma IRQ por parte de varios dispositivos en slots PCI, se introdujo un sistema de IRQ dinámicas que facultaba una asignación automática a través de un proceso de "negociación" entre la BIOS, el sistema operativo y el bus PCI. Un conflicto entre IRQs puede causar un mal funcionamiento del PC, una caída de rendimiento o la imposibilidad de usar determinados aparatos. Para comprobar el estado de los puertos y de las conexiones es posible descargar programas (gratis o de pago) de análisis IRQ y de ayuda para diagnosticar problemas de IRQ y de su asignación.

IOCTL

ioctl es una llamada de sistema en Unix que permite a una aplicación controlar o comunicarse con un driver de dispositivo, fuera de los usuales read/write de datos. Esta llamada se originó en la versión 7 del AT&T Unix. Su nombre abrevia la frase input/output control.

Una llamada ioctl toma como parámetros:

1. un descriptor de archivos abierto
2. un número de código de requerimiento
3. también un valor entero, posiblemente sin signo (va al driver), o un puntero a datos (también va al driver y vuelve de él)

El kernel generalmente envía un ioctl directamente al driver, el cual puede interpretar el número de requerimiento y datos en cualquier forma requerida. Los escritores del driver documentan cada número de requerimiento del driver para ese driver particular, y los proveen de constantes en el archivo de cabeceras (*.h)

Algunos sistemas tienen convenciones en su codificación, dentro del número de codificación, el tamaño de los datos a ser transferidos desde o hacia el driver del dispositivo, la dirección de la transferencia de datos y la identidad del driver implementando el requerimiento.

Independientemente de que esa convención se cumpla o no, el kernel y el driver colaboran para entregar un código de error uniforme (señalados por la constante simbólica ENOTTY) a una aplicación que haga un requerimiento al driver que no reconozca.

El nemónico ENOTTY (tradicionalmente asociado con el mensaje textual "Not a typerwriter") viene del hecho de que en los sistemas iniciales que incorporaba una llamada ioctl, solo el dispositivo teletipo (tty) planteaba este error. A través del nemónico simbólico es ajustado por los requerimientos de compatibilidad; algunos sistemas modernos muy útilmente prestan un mensaje más general, como: "Inappropriate device control operation", o la localización del mismo.

TCSETS ejemplifica un ioctl en un puerto serial. Las llamadas de lectura y escritura normales en un puerto serial reciben y envían paquetes de datos. Una llamada ioctl(fd, TCSETS, data), independiente de las llamadas normales I/O, controla varias opciones del controlador, como la manipulación de caracteres especiales, o las señales de salida en el puerto (por ejemplo, la señal DTR).

MÓDULOS

Un módulo del kernel es un fragmento de código o binarios que pueden ser cargado y eliminados del kernel según las necesidades de este. Tienen el objetivo de extender sus funcionalidades son fragmentos de código que pueden ser cargados y eliminados del núcleo bajo demanda. Extienden la funcionalidad del núcleo sin necesidad de reiniciar el sistema.

Esto es gracias a que el kernel tiene un diseño modular, cuando se instala un nuevo componente o se inicia la computadora los módulos son cargados de forma dinámica para que funcionen de forma transparente.

Los módulos son almacenados en /lib/modules/nombre_del_kernel. Para saber su versión de kernel puede emplear la instrucción uname -r .

MEMORIA COMPARTIDA

En informática, la memoria compartida es aquel tipo de memoria que puede ser accedida por múltiples programas, ya sea para comunicarse entre ellos o para evitar copias redundantes. La memoria compartida es un modo eficaz de pasar datos entre aplicaciones. Dependiendo del contexto, los programas pueden ejecutarse en un mismo procesador o en procesadores separados. La memoria usada entre dos hilos de ejecución dentro de un mismo programa se conoce también como memoria compartida.]

SOFTWARE

En el software, memoria compartida puede referirse a:

- Un método de comunicación entre procesos, por ejemplo: el intercambio de datos entre dos programas ejecutándose al mismo tiempo. Uno de los procesos creará un área en RAM a la que el otro pueda acceder, o
- Un método para conservar espacio en la memoria, usando mapeos virtuales o bien soporte explícito del programa en cuestión, dirigiendo los accesos a una sola instancia de datos que normalmente serían duplicados. Comúnmente destinado para bibliotecas de enlace dinámico dinámicas y el espacio de usuario (XIP, "execute in place").

Dado que ambos procesos pueden acceder al área de memoria compartida como memoria de trabajo regular, esta es una forma de comunicación veloz (al contrario de otros mecanismos de comunicación entre procesos como tuberías nombradas, socket de dominio UNIX o CORBA. En cambio, este sistema es menos potente, si, por ejemplo, los procesos que se comunican deben ejecutarse en la misma máquina (en cuanto a otros métodos de comunicación entre procesos,

solo los sockets del Internet Domain (no los sockets de UNIX), pueden usar una red de computadoras). Esto se debe a que se requiere mayor atención y cuidado si los procesos que comparten memoria corren en CPUs separadas y la arquitectura subyacente no soporta coherencia de caché.

La comunicación entre procesos por memoria compartida se usa en UNIX, para transferir imágenes entre una aplicación y un XServer, o en la biblioteca COM de Windows dentro del objeto IStream devuelto por la función CoMarshalInterThreadInterfaceInStream. Las bibliotecas de enlace dinámico se copian una sola vez en la memoria y son "mapeadas" para múltiples procesos. Sólo las páginas que estén personalizadas se duplican, normalmente con un mecanismo conocido como copy-on-write que de manera transparente copia la página cuando se intenta escribir en ésta, y después permite que la escritura se realice en la copia privada.

Hardware:

Se refiere a la situación dónde cada nodo (computadora) comparte sus tablas de páginas, su memoria virtual y todo lo que habita en la memoria RAM con otros nodos en la misma red de interconexión. Un ejemplo de memoria compartida es DSM.

En otras palabras, una computadora convencional tiene una memoria RAM y un CPU (entre otras partes de hardware), las cuales están comunicadas de manera local. Cuando se habla de multicomputadoras es necesario entenderlas como un conjunto de máquinas convencionales trabajando coordinadamente con el objetivo de compartir recursos de software y hardware para tareas muy específicas, como por ejemplo la investigación. El hecho de que cada CPU sólo pueda acceder a la RAM local no beneficia mucho y agrega complejidad a los programadores de

multicomputadoras, con la compartición de memoria se mitiga esta complejidad ya que todas las CPUs pueden acceder a la misma RAM.

INSMOD

El comando insmod en los sistemas Linux se usa para insertar módulos en el kernel. Linux es un sistema operativo que permite al usuario cargar módulos del kernel en tiempo de ejecución para ampliar las funcionalidades del kernel. Los LKM (módulos de núcleo cargables) se utilizan generalmente para agregar soporte para nuevo hardware (como controladores de dispositivos) y / o sistemas de archivos, o para agregar llamadas al sistema. Este comando inserta el archivo de objeto del kernel (.ko) en el kernel con / sin argumentos, junto con algunas opciones adicionales.

Sintaxis:

```
insmod [nombre de archivo] [módulo-opciones ...]
```

RMMOD

Linux rmmod comando para extraer el módulo.

la ejecución del comando rmmod, puede quitar módulos innecesarios. núcleo del sistema operativo Linux tiene un carácter modular, esto debería ser al kernel en tiempo de compilación, aconsejó a poner todas las funciones se colocan como el núcleo. Puede utilizar estas funciones compilan en un módulo separado, y luego se va a cargarse cuando se necesitan.

gramática

```
rmmod [-as][XXX...]
```

parámetros:

- **-a** Eliminar todos los módulos innecesarios actualmente.

- **-s** la información al servicio permanente syslog, interfaz en lugar terminal.

ASSEMBLER

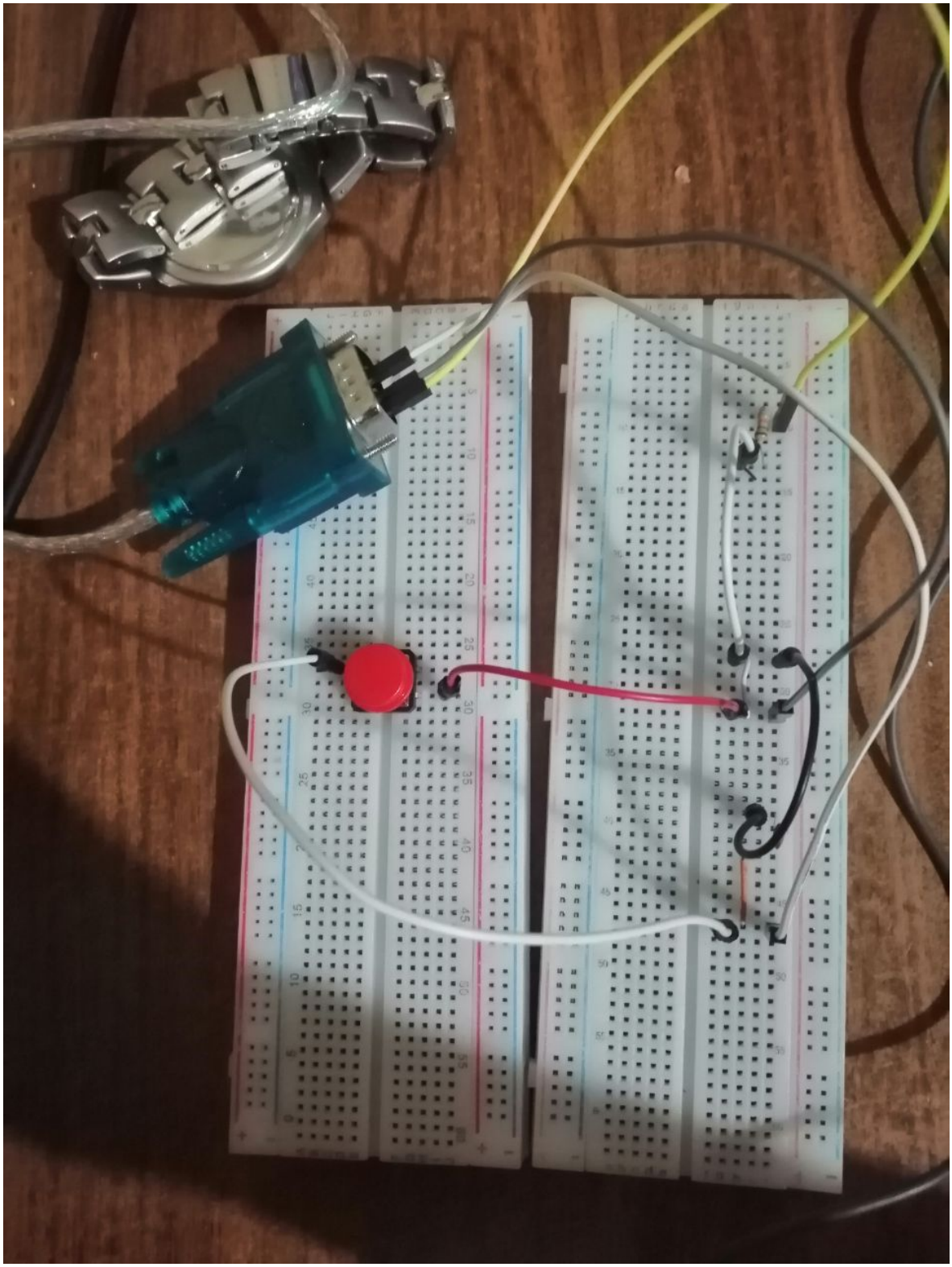
El lenguaje ensamblador o assembly (en inglés: assembly language y la abreviación asm) es un lenguaje de programación de bajo nivel. Consiste en un conjunto de mnemónicos que representan instrucciones básicas para los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura de procesador y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador. Cada arquitectura de procesador tiene su propio lenguaje ensamblador que usualmente es definida por el fabricante de hardware, y está basada en los mnemónicos que simbolizan los pasos de procesamiento (las instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje. Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual). Esto está en contraste con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portables.

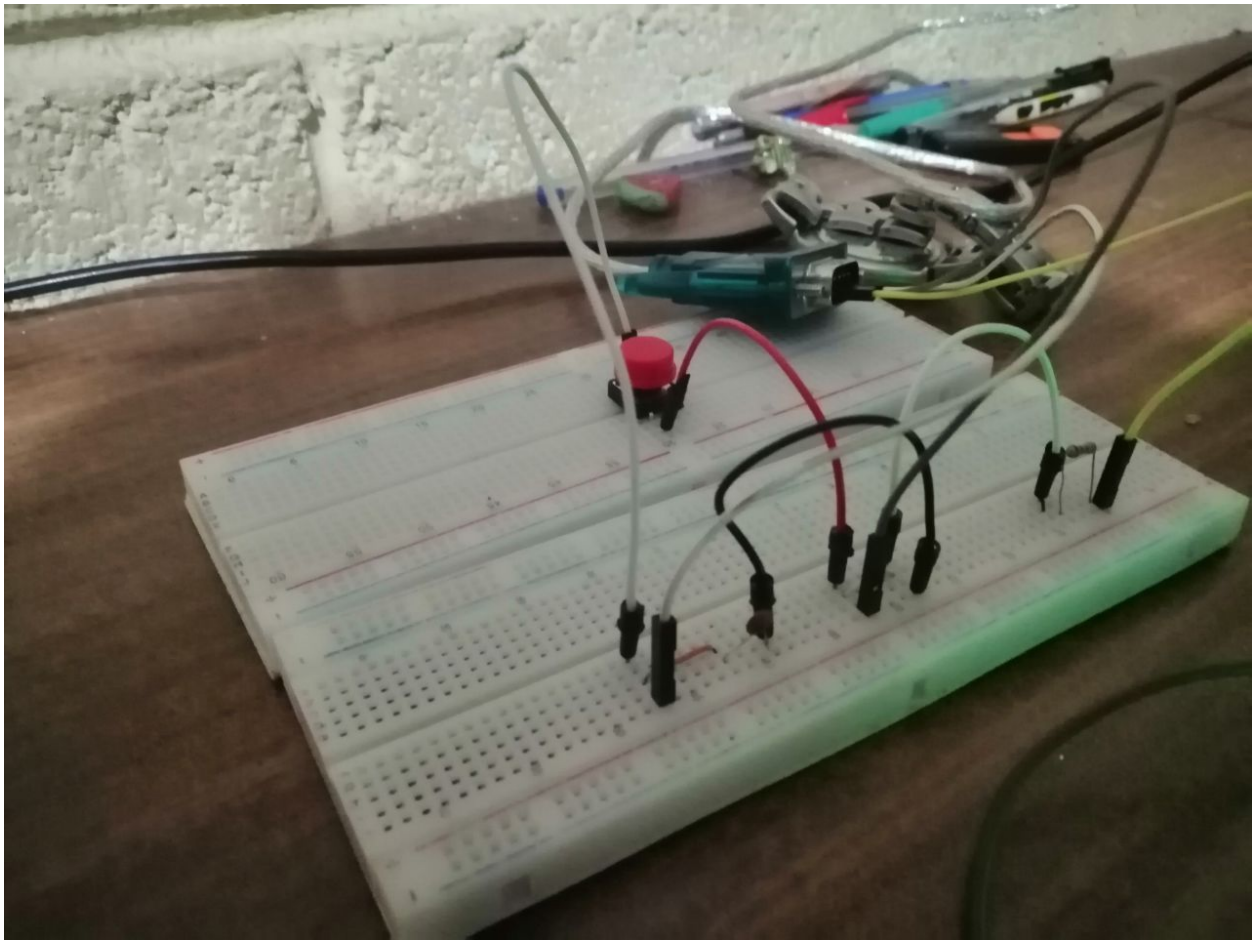
Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina.

Muchos sofisticados ensambladores ofrecen mecanismos adicionales para facilitar el desarrollo del programa, controlar el proceso de ensamblaje, y la ayuda de depuración. Particularmente, la mayoría de los ensambladores modernos incluyen una facilidad de macro (descrita más abajo), y se llaman macro ensambladores.

Fue usado principalmente en los inicios del desarrollo de software, cuando aún no se contaba con potentes lenguajes de alto nivel y los recursos eran limitados. Actualmente se utiliza con frecuencia en ambientes académicos y de investigación, especialmente cuando se requiere la manipulación directa de hardware, alto rendimiento, o un uso de recursos controlado y reducido. También es utilizado en el desarrollo de controladores de dispositivo (en inglés, device drivers) y en el desarrollo de sistemas operativos, debido a la necesidad del acceso directo a las instrucciones de la máquina. Muchos dispositivos programables (como los microcontroladores) aún cuentan con el ensamblador como la única manera de ser manipulados.

CREACIÓN DEL CIRCUITO





CODIFICACIÓN

Envio:

```
//tiempo de la ultima pulsacion
struct tm tm;
strptime(memo2, "%H:%M:%S", &tm);
time_t t = mktime(&tm);
//tiempo actual
int sec = ts->tm_sec - tm.tm_sec;
printf("Execution time = %d\n", sec);
if(sec >= rango || sec < 0){
        conteo = 1;
} else {
        int flag = 0;
        for(int a = 0; a < rango; a++){
                char zz[10];
                sprintf(zz,"%d",conteo);
                if(strcmp(memo1, zz) == 0){
                        conteo = conteo +1;
                        flag = 1;
                        break;
                }
        }
        if(flag == 0){
                conteo = 1;
        }
}
```

Recibo:

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<sys/shm.h>

int main(){
        int id;
        void *mem;
        while(1){
                id=shmget((key_t)2345, 1024, 0666);
                if(id!=-1){
                        mem = shmat(id,NULL,0);
                        char* temp = (char *) mem;
                        if(temp == "1"){
                                printf("He sido pulsado %s vez\n",temp);
                        } else {
                                printf("He sido pulsado %s veces\n",temp);
                        }
                } else {
                        printf(" ");
                }
        sleep(1);
        }
```

BIBLIOGRAFÍA

https://es.wikipedia.org/wiki/Memoria_compartida

https://www.geeksforgeeks.org/insmod-command-in-linux-with-examples/

http://www.w3big.com/es/linux/linux-comm-rmmod.html

https://www.monografias.com/docs112/gestion-interrupciones-sistemas-operativos/gestion-interrupciones-sistemas-operativos2.shtml

https://www.sciencedirect.com/topics/computer-science/programmable-interrupt-controller

https://forums.sifive.com/t/beginner-trying-to-set-up-timer-irq-in-assembler-how-to-print-csrs-in-gdb/2764

https://es.wikipedia.org/wiki/Lenguaje_ensamblador