

Arquitectura de computadoras

Trabajo Práctico N°4 - 2024

Entrada Salida. Interrupciones.

OBJETIVOS

- Comprender y diferenciar los mecanismos de entrada-salida a través de mapeo en memoria e instrucciones especiales.
- Programar interrupciones simples para detectar eventos asíncronos.
- Utilizar funciones adicionales que poseen los sistemas embebidos.
- Entender la diferencia entre un lenguaje de alto nivel y ensamblador.

Metodología

Trabajo grupal. Número de estudiantes por grupo según computadoras disponibles.
Tiempo de realización estimado: 3 clases.

Aprobación

- Mostrar en clases los programas que se solicitan en las actividades 1 y 2 funcionando.
- Subir a la plataforma Moodle los programas funcionando correctamente.
- Responder el cuestionario “Trabajo Práctico N°4 - Cuestionario” luego de realizar la actividad 3.

Materiales y equipamientos necesarios:

- Computadoras (Notebook, PC de escritorio, etc).
- Simulador Assembler Simulator de 16-bit:
<https://parraman.github.io/asm-simulator/>
- Simulador de placa de desarrollo Arduino UNO a través de un classroom de Tinkercad. Ver instrucciones en el Anexo 3.

Actividad 1: Entrada Salida en Simulador de 16 bits

En esta actividad utilizará el simulador Assembler Simulator de 16-bit: (<https://parraman.github.io/asm-simulator/>). En dicho simulador se mapea:

- Un display de 32 caracteres desde la posición 0x02E0 hasta la posición 0x02FF. Cada posición mapea un carácter en el display. Los caracteres se escriben en código ASCII.
- Una pantalla de 16x16 píxeles de 255 colores se mapea en memoria desde la posición 0x0300 hasta la posición 0x03FF. Cada posición de memoria en dicho rango mapea un píxel de la pantalla.

Por otro lado, el simulador incluye un teclado que puede accederse a través de instrucciones especiales IN y OUT.

Actividad 1.1: Uso de Teclado y Pantalla.

Escriba un programa que pinte píxeles de la pantalla comenzando por el píxel superior izquierdo (dirección 0x0300) y termine en el píxel inferior derecho (dirección 0x03FF). El programa deberá comenzar pintando los píxeles de color azul. Al apretar la tecla 2 del teclado del simulador, deberá cambiar el color de pintado por amarillo. Al apretar la tecla 1 del teclado, deberá cambiar el color de pintado por azul.

Actividad 1.2: Mario y Luigi

a) Abra el ejemplo “Draw Sprite” (Para ello, vaya a “File”, luego a “Samples”, luego a “Draw Sprite”).

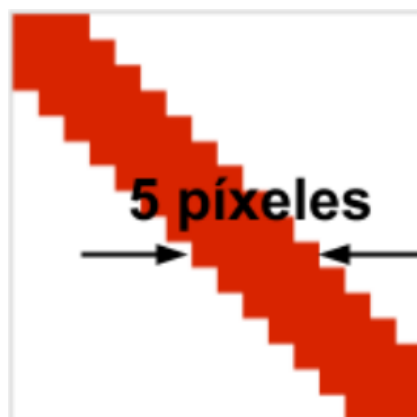
b) Ensamble y ejecute el programa. El mismo dibuja un “Mario Bross” por pantalla.

c) Cambie el código de manera que luego de dibujar el Mario Bross, cambie el Mario Bross por Luigi cambiando el color del traje rojo por uno verde. El cambio de color debe realizarse a nivel de código, no a nivel de constantes.

Sugerencia: Quite la instrucción HLT. En su lugar, cree un bucle que lea el color de cada píxel de la pantalla, desde el primero (0x0300) al último (0x03FF). Si el color de un pixel es rojo (0xC4), cámbielo por verde (0x15). Utilice instrucciones de comparación (CMP) y saltos condicionales e incondicionales (JZ, JNZ, JMP).

1.3 Camiseta de River Plate

Escriba un programa que dibuje en la pantalla una franja roja a -45° sobre fondo blanco (similar a la camiseta del club atlético River Plate), como se muestra en la figura.



El programa deberá cumplir las siguientes condiciones:

- a) La franja debe tener 5 píxeles de ancho.
- b) La franja debe estar centrada

Actividad 2 - Interrupciones con Arduino

En esta actividad se utilizará un simulador de Arduino con placa de desarrollo.

Actividad 2.1: Introducción a interrupciones

1) Utilizando la interfaz de desarrollo de las plataformas Arduino, escriba un programa que encienda los led del 6 al 13 consecutivamente, uno a la vez, durante 0,8 segundos cada led. Al terminar, deberá comenzar nuevamente, encendiendo desde el led 6.

Importante!: primero configure los pines respectivos como salidas.

Sugerencia 1: Cree primero un programa que encienda y apague un solo led, luego de verificar que funciona correctamente, extienda el código a todos los leds.

Sugerencia 2: Utilice la función `delay(tiempo)` para implementar una pausa de 0.8 segundos, donde el argumento tiempo es el tiempo en milisegundos que dura la pausa.

2) Implemente una rutina de servicio para la interrupción del pin 2 (Necesitará utilizar la primitiva `attachInterrupt`". En el anexo 1 encontrará instrucciones de uso). La rutina de servicio deberá hacer que el led 7 parpadee durante 4 segundos, siendo el periodo de parpadeo de 0.1 segundos (0.05 segundos encendido, 0.05 segundos apagado), luego de los 4 segundos, el led 7 debe permanecer encendido durante un segundo completo, para luego salir de la rutina de servicio. Los demás leds deben permanecer apagados.

Importante!: primero configure el pin 2 como entrada.

Nota: La función `delay(tiempo)` puede no funcionar, o funcionar de manera diferente si la utiliza dentro de una rutina de servicio. Deberá utilizar la función `delayMicroseconds(tiempo)` donde tiempo se expresa en microsegundos. Deberá considerar que el valor máximo que puede tomar el argumento tiempo de la función `delayMicroseconds(tiempo)` es 16383 (0.016 seg).

3) Implemente una rutina de servicio para la interrupción del pin 3. La rutina de servicio deberá hacer que el led 12 parpadee durante 4 segundos, siendo el periodo de parpadeo de 0.1 segundos (0.05 segundos encendido, 0.05 segundos apagado), luego de los 4 segundos, el led 12 debe permanecer encendido durante un segundo completo, para luego salir de la rutina de servicio. Los demás leds deben permanecer apagados.

Importante!: primero configure el pin 3 como entrada.

4) Analice experimentalmente: Pulse los pulsadores en secuencias rápidas tales como:

- 3, 3, 2
- 2, 3, 2
- 3, 2, 3, 2

- 3, 2, 2, 2, 2, 2

y analice el comportamiento del programa. En base a lo observado, conteste las siguientes preguntas (a través del cuestionario “Cuestionario Trabajo Práctico N°4 - 2024” que encontrará en la plataforma Moodle):

- a - ¿Permite el microprocesador anidamiento de interrupciones?
- b - ¿Cuál interrupción tiene mayor prioridad?
- c - ¿Por qué la función `delay(time)` funciona adecuadamente dentro del código principal, pero funciona de manera diferente, o directamente no funciona, dentro de una rutina de servicio?
- d - ¿Qué ocurriría con las situaciones descritas en el puntos c si el microprocesador permitiera interrupciones anidadas?
- e - Si presiona los pulsadores siguiendo la secuencia 3, 2, 2, 2, 2, 2 rápidamente, ¿Se ejecutará primero la rutina de servicio del pulsador 3, y luego se ejecutará 5 veces la rutina de servicio del pulsador 2?

Actividad 2.2 - Funciones adicionales en sistemas embebidos

Los pines, además de las funciones `digitalRead()` y `digitalWrite()`, tienen muchas otras funciones asociadas. Veremos una de ellas, medición del ancho de un pulso con “`unsigned long pulseIn(pin,nivel,timeout)`”, y utilizaremos esta función para medir distancia utilizando el sensor de distancia HC-SR04.

La función “`pulseIn(pin,nivel,timeout)`” mide el ancho de un pulso aplicado a un pin, donde:

- pin: pin en el cual queremos medir el ancho de un pulso.
- nivel: puede valer:
 - HIGH si el pulso es un pulso con nivel de tensión alto.
 - LOW si el pulso es un pulso con nivel de tensión bajo.



Pulso alto o HIGH



Pulso bajo o LOW

- Timeout: tiempo máximo en microsegundos que el procesador espera el pulso. Si después de este tiempo no se aplicó ningún pulso, el procesador interrumpe la ejecución de la función `pulseIn(pin,nivel,timeout)`.

El sensor de distancia HC-SR04 funciona de la siguiente manera:

- a. Se aplica un pulso HIGH de al menos 10 microsegundos en el pin “Trig”.

- b. Se espera un pulso HIGH en el pin “Echo”. El ancho de dicho pulso en microsegundos es proporcional a la distancia. Para conocer la distancia en centímetros, se debe dividir por 58 (las señal de ultrasonido recorre un centímetro en 58 microsegundos).

Actividad a realizar:

Esta actividad puede realizarse en conjunto con la actividad 2.1 (Agregando nuevo código al ya implementado).

1. Conecte los pines de alimentación del sensor a +5V y GND, y los pines “Trig” y “Echo” a pines de entrada/salida del Arduino, por ejemplo el pin 5 para “Trig” y el pin 4 para “Echo”.
2. Configure estos pines como corresponde, el pin donde esté conectado “Trig” debe ser salida, y el pin donde está conectado “Echo” debe ser entrada.
3. Escriba un programa que genere un pulso en el pin “Trig” como el requerido.
4. Utilice la función “unsigned long pulseIn(pin,nivel,timeout)” para leer el ancho del pulso. Imprima la información de la distancia por pantalla (recuerde dividir el valor leído por 58).
5. Escriba un programa que implemente una alarma de distancia que realice las siguientes tareas:
 - a. Mida la distancia cada 0.8 segundos.
 - b. Al presionar el pulsador 2 la alarma debe activarse.
 - c. Al presionar el pulsador 3 la alarma debe desactivarse.
 - d. Se debe mostrar por pantalla la frase “Alarma Activada” o “Alarma Desactivada”, según el caso.
 - e. Por cada lectura de distancia, debe mostrarse el valor de la distancia y la leyenda “alarma activada” o “alarma desactivada” según corresponda.
 - f. Si la alarma está activada y se detecta que la distancia del intruso es menor a 1 metro, todos los leds deben parpadear rápidamente, y se debe mostrar por pantalla un mensaje de alerta de intruso.
 - g. Si la alarma está desactivada y se detecta que la distancia del intruso es menor a 1 metro, no debe mostrarse nada ni realizar ninguna acción, ya que la alarma está desactivada.

Anexo 1: Escribiendo programas en Arduino

Estructura básica de un programa en el IDE de Arduino:

```
void setup() {  
  /*Dentro de la función setup se implementa el código que configura la plataforma.  
  Acá deben configurarse pines como entrada/salida, periféricos a utilizar, habilitar  
  interrupciones y declarar el nombre de las respectivas rutinas de servicio. La función  
  setup solo se ejecuta una vez al iniciar la plataforma.*/  
}  
void loop() {  
  /*La función loop se ejecuta repetidamente sin fin. Acá debe colocar el código que  
  se ejecutará repetidamente mientras la plataforma esté encendida*/  
}  
void función(){  
  /*Acá puede declarar funciones y rutinas de servicio de interrupciones.*/  
}  
void rutina_de_servicio(){  
  /*Una rutina de servicio se implementa como una función que será llamada  
  automáticamente cuando ocurra la interrupción. Deberá primero habilitar la  
  interrupción en la función setup() y declarar que función implementa su rutina de  
  servicio, mediante la instrucción:  
  attachInterrupt(número de la fuente de interrupción, nombre de la rutina de servicio,  
  evento que dispara la interrupción);  
  Pin 2: Interrupción número 0  
  Pin 3: Interrupción número 1  
  Eventos que disparan las interrupciones:  
  LOW: para activar la interrupción cuando el pin es bajo.  
  CHANGE: para activar la interrupción cuando el pin cambia de valor.  
  RISING: dispara la interrupción cuando se detecta un flanco de subida.  
  FALLING: dispara la interrupción cuando se detecta un flanco de bajada. */  
}
```

Funciones útiles en C para Arduino:

```
pinMode(13, OUTPUT); //declara en pin 13 como salida. No retorna nada.  
pinMode(3, INPUT); //declara el pin 3 como entrada. No retorna nada.  
digitalRead(pin); // Lee el valor del pin. Retorna un entero que puede ser 1 o 0.  
digitalWrite(pin, value); //Escribe value en el pin indicado, value puede valer HIGH o  
LOW. Serial.begin(9600); //Inicializa el puerto serie.
```

`Serial.println(datos a imprimir);`//imprime datos en el puerto serial. Agrega un salto de línea al final.

`Serial.print(datos a imprimir);`//imprime datos en el puerto serial.

Fuente: <https://www.arduino.cc/reference/en/#functions>

Anexo 2: Obteniendo información sobre una computadora

Puede obtener información sobre una computadora siguiendo alguno de los siguientes procedimientos:

- En Linux:
 - Ejecute los programas `lshw`, `lspci` y `lsusb` en su computadora (si no los tiene instalados, puede descargarlos desde los repositorios de Linux).
 - Puede generar una salida fácilmente legible con “`sudo lshw -html > nombre_archivo.html`”. Este comando mostrará los resultados en un archivo html que será creado en su carpeta principal.
- EN Windows:
 - Programa “`dxdiag`”: Use la barra de búsqueda de programas y archivos para buscar y ejecutar la aplicación `dxdiag` (o presione Windows+R para abrir la herramienta “ejecutar”). Si se le pregunta si desea comprobar firmas digitales, seleccione no.
 - Herramienta “Administrador de dispositivos”. Podrá encontrarla en el Panel de Control (Menú Inicio de Windows -> Panel de Control).
 - Herramienta Información del Sistema. Use la barra de búsqueda de programas y archivos para buscar y ejecutar la aplicación Información del Sistema (o presione Windows+R para abrir la herramienta “ejecutar” y ejecute “`msinfo32`”. También puede encontrarla en Accesorios->Herramientas del sistema).

Anexo 3: Simulador de Arduino en Tinkercad

1. Registrarse en la clase “Arquitectura de computadoras 2024. Tp: Entrada salida”
 - a. Ingresar al siguiente link:
<https://www.tinkercad.com/joinclass/SNAKMKZCJ>
 - b. Regístrate en la plataforma mediante alguno de los métodos de autenticación permitidos. Se puede crear un usuario de la plataforma mediante un correo electrónico o realizar el login con cuentas de Google, Microsoft, Facebook, etc.
2. Acceder a la simulación:
 - a. Posteriormente y en una pestaña nueva, ingresar al siguiente link:
<https://www.tinkercad.com/things/1SwgX1ewGBS>
 - b. Copie y modifique el proyecto para adaptarlo a cada uno de los puntos del práctico. Repetir este paso para cada uno de los ejercicios. Renombrar cada uno de los proyectos con el número de ejercicio para que los docentes puedan controlar el avance de cada uno en este trabajo práctico.