Universidad Nacional de Cuyo
Facultad de Ingeniería
Licenciatura en Ciencias de la Computación

Trabajo Práctico Nº 4

Arquitectura de Computadoras

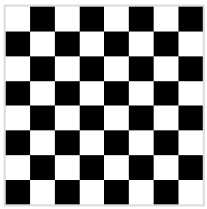
Entrada - Salida

2024

Adriano Santino Fabris
Gonzalo Padilla Lumelli
Mariano Robledo

Actividad 1 - Entrada salida mediante mapeo en memoria e instrucciones especiales - Lenguaje ensamblador

En esta actividad utilizará el simulador Assembler Simulator de 16-bit: (https://parraman.github.io/asm-simulator/). En dicho simulador, 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 la instrucción especial IN. Dicha instrucción deposita en el registro A el valor leído desde el teclado.

1.1 Tablero de ajedrez

Escriba un programa que dibuje en la pantalla un tablero tipo "tablero de ajedrez.ªlternando cuadros blancos con cuadros negros, como se muestra en la figura.

El programa deberá cumplir las siguientes condiciones:

a) Cada cuadro debe tener un tamaño de 2x2 pixeles.

b) Al presionar la tecla 1, debe borrarse todo el tablero.

c) Al presionar la tecla 2, debe redibujar todo el tablero.

Solución

```
ScrStart  EQU 0x0300 ; Comienzo de la pantalla
ScrEnd    EQU 0x0400 ; Fin de la pantalla
KeyStatus EQU 0x0005 ; Registro de estado del teclado
KeyData   EQU 0x0006 ; Registro de botón presionada

Input:              ; Loop que realiza el polling de los botones
    IN KeyStatus    ; Estado del teclado a registro A
    CMP A, 0
    JZ Input        ; Si se presionó, seguimos, si no, volvemos a 'Input'
    MOV D, ScrStart ; Dirección de la pantalla al registro D
    IN KeyData      ; Botón presionado al registro A
    CMP A, 0x032    ; Comparamos con el carácter ASCII de '2'
    JZ Print        ; Si es '2', saltamos a la subrutina 'Print'
    CMP A, 0x031    ; Comparamos con el carácter ASCII de '1'
    JZ Clear        ; Si es '1', saltamos a la subrutina 'Clear'
    JMP Input       ; De otra forma, volvemos al loop de 'Input'

Clear:              ; Subrutina para borrar el tablero
    MOVB [D], 0xFF  ; Pintamos de blanco el pixel actual
    INC D           ; Incrementamos el puntero
    CMP D, ScrEnd   ; Verificar si llegamos al final
    JZ Input        ; Si es así, volver a 'Input'
    JMP Clear       ; Si no, volvemos a 'Clear' para seguir borrando
```

```
Print:              ; Subrutina que dibuja el tablero
    MOVB AL, 0      ; Color inicial a A
Row:                ; Loop de las 'filas' de 32 píxeles
    MOV B, 32       ; 32 al registro B
    NOTB AL         ; Invertimos el color
Square:             ; Loop de los 'cuadro' de 2 píxeles
    MOV C, 2        ; 2 al registro C
    NOTB AL         ; Invertimos el color
Pixel:              ; Loop para pintar píxeles
    MOVB [D], AL    ; Pintamos el pixel actual con el color actual
    INC D           ; Incrementamos el puntero
    CMP D, ScrEnd   ; Verificar si llegamos al final
    JZ Input        ; Si es así, volver a 'Input'
    DEC B           ; Decrementamos contador de 'fila'
    JZ Row          ; Si terminó la fila, volvemos a 'Row'
    DEC C           ; Decrementamos contador de 'cuadro'
    JZ Square       ; Si terminó el cuadro, volvemos a 'Square'
    JMP Pixel       ; Volver a 'Pixel'
```

Leemos las teclas presionadas mediante entrada controlada por programa. Dependiendo de la tecla presionada, saltamos a la subrutina 'Clear', que borra el tablero, o 'Print', que lo dibuja. Para dibujarlo se va recorriendo la pantalla pixel por pixel, cambiando de color cada 2 píxeles. Además, cada 32 píxeles, que corresponde a dos filas de píxeles, o una fila de cuadros, hacemos que se repita el color en el siguiente cuadro.

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 Bros" por la pantalla.

c) Modifique el código de manera que en lugar de Mario Bros, el programa dibuje a Luigi (Luigi tiene traje verde en lugar de traje rojo). El cambio de color debe realizarse a nivel de código, no a nivel de constantes.

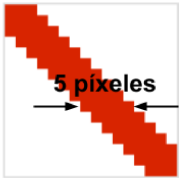
Solución

```
    JMP boot
vslDisplay EQU 0x300

sprite:
    DB "\xFF\xFF\xFF\xFF\xFF\xC4\xC4\xC4"
    DB "\xC4\xC4\xFF\xFF\xFF\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\xFF\xC4\xC4\xC4\xC4"
    DB "\xC4\xC4\xC4\xC4\xC4\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\xFF\x8C\x8C\x8C\xF4"
    DB "\xF4\x8C\xF4\xFF\xFF\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\x8C\xF4\x8C\xF4\xF4"
    DB "\xF4\x8C\xF4\xF4\xF4\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\x8C\xF4\x8C\x8C\xF4"
    DB "\xF4\xF4\x8C\xF4\xF4\xF4\xFF\xFF"
    DB "\xFF\xFF\xFF\x8C\x8C\xF4\xF4\xF4"
    DB "\xF4\x8C\x8C\x8C\x8C\xFF\xFF\xFF"
```

```
    DB "\xFF\xFF\xFF\xFF\xFF\xF4\xF4\xF4"
    DB "\xF4\xF4\xF4\xF4\xFF\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\xFF\x8C\x8C\xC4\x8C"
    DB "\x8C\x8C\xFF\xFF\xFF\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\x8C\x8C\x8C\xC4\x8C"
    DB "\x8C\xC4\x8C\x8C\x8C\xFF\xFF\xFF"
    DB "\xFF\xFF\x8C\x8C\x8C\x8C\xC4\xC4"
    DB "\xC4\xC4\x8C\x8C\x8C\x8C\xFF\xFF"
    DB "\xFF\xFF\xF4\xF4\x8C\xC4\xF4\xC4"
    DB "\xC4\xF4\xC4\x8C\xF4\xF4\xFF\xFF"
    DB "\xFF\xFF\xF4\xF4\xF4\xC4\xC4\xC4"
    DB "\xC4\xC4\xC4\xF4\xF4\xF4\xFF\xFF"
    DB "\xFF\xFF\xF4\xF4\xC4\xC4\xC4\xC4"
    DB "\xC4\xC4\xC4\xC4\xF4\xF4\xFF\xFF"
    DB "\xFF\xFF\xFF\xFF\xC4\xC4\xC4\xFF"
    DB "\xFF\xC4\xC4\xC4\xFF\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\x8C\x8C\x8C\xFF\xFF"
    DB "\xFF\xFF\x8C\x8C\x8C\xFF\xFF\xFF"
    DB "\xFF\xFF\x8C\x8C\x8C\x8C\xFF\xFF"
    DB "\xFF\xFF\x8C\x8C\x8C\x8C\xFF\xFF"

boot:
    MOV C, sprite       ; Carga al registro C la imagen.
    MOV D, vslDisplay   ; Puntero de la pantalla

.loop:
    MOVB AL, [C]        ; Mueve al registro A el valor del color actual del valor
↪   de C
    CMP A, 0xC4         ; Comparar con el color rojo
    JZ .changeColor     ; Si es rojo, saltamos a changeColor
.paint:
    MOVB [D], AL        ; Coloca el pixel en la posición de la pantalla actual.
    INC C
    INC D
    CMP D, 0x400        ; Verificamos si llegamos al final de la pantalla
    JNZ .loop           ; Si no, volvemos al loop
    HLT

.changeColor:
    MOVB AL, 0x10       ; Color verde al registro AL
    JMP .paint          ; Saltamos a .paint, ahora con el color verde
```

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.



a) La franja debe tener 5 píxeles de ancho.

b) La franja debe estar centrada.

Solución

```
MOVB AL, 0x0002     ; ancho de cada línea horizontal que conforma la franja es de
↪  longitud 2
MOVB BH, 0x03
MOVB CH, 0x00

Secuencia2:         ; bucle para dibujar la franja diagonal
MOVB CL, 0x00       ; pinta de blanco
MOVB [B], 0x0C4     ; pinta de rojo
CMPB BL, 255        ; comprueba el fin de la pantalla
JZ Fin              ; termina el programa
INCB BL             ; incrementa BL para movernos horizontalmente
CMPB BL, 240
JZ Salto2
INCB AL             ; incrementa el ancho de la franja
CMPB AL, 0x0005     ; compara el ancho de franja con 5 (longitud deseada)
JZ Secuencia1       ; si es 5 debe reiniciarse el ancho de la franja
JMP Secuencia2


Secuencia1:         ; reinicia el grosor de la franja
MOVB AL, 0x00       ; reinicia AL a 0 para comenzar una nueva franja
INCB BL
CMPB BL, 0x000F     ; 15 es un posible límite horizontal
JZ Salto1
INCB CL             ; controla la posición vertical
CMPB CL, 0x000C     ; comparar CL con 12 un límite vertical específico
JZ Secuencia2       ; si CL es 12, salta de nuevo a Secuencia2 para comenzar una
↪  nueva iteración
JMP Secuencia1

Salto2:             ; controla de desplazamiento horizontal
INCB BL
JMP Secuencia1

Salto1:             ; ajusta el desplazamiento horizontal y grosor
INCB BL
INCB AL
MOVB [B], 0x00      ; establece el color blanco
JMP Secuencia2      ; salta a Secuencia2 para continuar el dibujo

Fin:
HLT                 ; finaliza el programa
```

Actividad 2 - Interrupciones con Arduino

Actividad 2.1 - Interrupciones de los pines 2 y 3

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.

Solución

```
void setup() {
  pinMode(2, INPUT);
  pinMode(7,OUTPUT);
}

int count=6;

void loop() {
  if (digitalRead(2) == HIGH){
    for(int i=0;i<=4;i+=1){
      if (i%2!=0){
        digitalWrite(7,HIGH);
        delay(0.01);
      }else{
        digitalWrite(7,LOW);
        delay(50);
      }
    }
    digitalWrite(7,HIGH);
    delay(1000);
  }
}
```

2. Escriba un bloque de código (puede ser como subrutina) que, al pulsar el pulsador conectado al pin 2, 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.

Solución

```
void setup() {
  pinMode(3, INPUT);
  pinMode(12,OUTPUT);
}

void loop() {
  if (digitalRead(3) == HIGH){
    for(int i=0;i<=40;i+=1){
      if (i%2!=0){
        digitalWrite(12,HIGH);
        delay(50);
      }else{
        digitalWrite(12,LOW);
        delay(50);
      }
    }
    digitalWrite(12,HIGH);
    delay(1000);
    digitalWrite(12,LOW);
    }
  }
```

3. Escriba un bloque de código (puede ser como subrutina) que, al pulsar el pulsador conectado al pin 3, 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.

Solución

```
void setup() {
  pinMode(3, INPUT);
  pinMode(12,OUTPUT);
}

int count=6;

void loop() {
  if (digitalRead(2) == HIGH){
    for(int i=0;i<=40;i+=1){
      if (i%2!=0){
        digitalWrite(12,HIGH);
        delay(50);
      }else{
        digitalWrite(12,LOW);
        delay(50);
      }
    }
    digitalWrite(12,HIGH);
    delay(1000);
    digitalWrite(12,LOW);
  }
}
```

4. Implemente los bloques de códigos escritos en los puntos 2 y 3 como rutinas de servicios de respectivas interrupciones, que deberán dispararse al pulsar los pulsadores 2 y 3 (Necesitará utilizar la primitiva “attachInterrupt”. En el anexo 1 encontrará instrucciones de uso).

Solución

```
void setup() {
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(7,OUTPUT);
  pinMode(12,OUTPUT);

  attachInterrupt(digitalPinToInterrupt(2), pulsador1, FALLING);
  attachInterrupt(digitalPinToInterrupt(3), pulsador2, FALLING);
}

void loop() {}

void pulsador1(){
  if (digitalRead(2) == HIGH){
    for(int i=0;i<=40;i+=1){
      if (i%2!=0){
        digitalWrite(7,HIGH);
        for(int j=0;j<=5;j++){
          delayMicroseconds(10000);
        }

      }else{
        digitalWrite(7,LOW);
        for(int j=0;j<=5;j++){
          delayMicroseconds(10000);
        }
      }
    }
    digitalWrite(7,HIGH);
    for(int j=0;j<=100;j++){
          delayMicroseconds(10000);
        }
    digitalWrite(7,LOW);
  }
}

void pulsador2(){
  if (digitalRead(3) == HIGH){
    for(int i=0;i<=40;i+=1){
      if (i%2!=0){
        digitalWrite(12,HIGH);
        for(int j=0;j<=5;j++){
          delayMicroseconds(10000);
        }
      }else{
        digitalWrite(12,LOW);
        for(int j=0;j<=5;j++){
          delayMicroseconds(10000);
        }
      }
```

```
    }
    digitalWrite(12,HIGH);
    for(int j=0;j<=100;j++){
          delayMicroseconds(10000);
        }
    digitalWrite(12,LOW);
  }
}
```

5. 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 - 2023" 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?

Solución

Contestado a través del cuestionario en la plataforma Moodle.

Actividad 2.2 - Funciones adicionales en sistemas embebidos

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 0.8 segundos.

   b) Al presionar el pulsador 2 y mantenerlo presionado durante más de 5 segundos, la alarma debe activarse o desactivarse, según el estado previo (si estaba activada, debe desactivarse. Si estaba desactivada, debe activarse). Se debe mostrar por pantalla la frase "Alarma Activada" o "Alarma Desactivada", según el caso.

   c) Por cada lectura de distancia, debe mostrarse el valor de la distancia y la leyenda "alarma activada" o "alarma desactivada" según corresponda.

   d) 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.

   e) 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.

Solución

```
const int TRIGGER_PULSE_TIME = 15;
const int TRIGGER_PIN = 5;
const int ECHO_PIN = 4;
const int ALARM_TOGGLE_PIN = 2;
const int ALARM_CALC_INTERVAL = 800;
const int ACTIVATION_DISTANCE = 100;

bool alarmActivated = false;
bool buttonPressed = false;
unsigned long timePressed = 0;
bool buttonActive = true;
bool LEDsOn = false;

void setup() {
    Serial.begin(9600);
    pinMode(TRIGGER_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    pinMode(ALARM_TOGGLE_PIN, INPUT);

    for(int pin = 6; pin <= 13; pin++)
        pinMode(pin, OUTPUT);

    buttonPressed = digitalRead(ALARM_TOGGLE_PIN);

    attachInterrupt(digitalPinToInterrupt(ALARM_TOGGLE_PIN),
                    buttonToggleRoutine,
                    CHANGE);
}

void buttonToggleRoutine() {
    buttonActive = true;
    buttonPressed = !buttonPressed;
    if (buttonPressed)
        timePressed = millis();
}
```

```
void loop() {
    handleInput();

    float distance = getDistance();

    Serial.print(distance, 1);
    Serial.print(" cm. ");

    if (alarmActivated) {
        Serial.println("Alarma activada.");
        if (distance && distance < ACTIVATION_DISTANCE) {
        Serial.println("INTRUSO DETECTADO!");
        toggleLEDs();
        }
        else
        turnOffLEDs();
    }
    else
        Serial.println("Alarma desactivada.");

    delay(ALARM_CALC_INTERVAL);
}

void handleInput() {
    if (buttonPressed && buttonActive) {
        unsigned long currentTime = millis();
        if (currentTime - timePressed > 5000) {
        alarmActivated = !alarmActivated;
        buttonActive = false;
        turnOffLEDs();
        }
    }
}

float getDistance() {
    digitalWrite(TRIGGER_PIN, LOW);
    delayMicroseconds(TRIGGER_PULSE_TIME);
    digitalWrite(TRIGGER_PIN, HIGH);
    delayMicroseconds(TRIGGER_PULSE_TIME);
    digitalWrite(TRIGGER_PIN, LOW);

    return pulseIn(ECHO_PIN, HIGH, 350 * 58) / 58.0;
}

void toggleLEDs() {
    LEDsOn = !LEDsOn;
    for(int i = 6; i <= 13; i++)
        digitalWrite(i, LEDsOn);
}

void turnOffLEDs() {
    LEDsOn = false;
    for(int i = 6; i <= 13; i++)
        digitalWrite(i, LOW);
}
```