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

Trabajo Práctico Nº 4

Arquitectura de Computadoras

Entrada - Salida

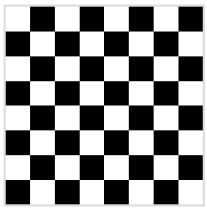
2024

Adriano Santino Fabris
Gonzalo Padilla Lumelli
Mariano Robledo

Mayo 2024

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

```
Pulsador:           ; Subrutina que realiza el polling de los botones
    IN 6            ; Cargamos el registro con el botón presionado
    MOV B,0x300     ; Cargamos en el resgistro B la dirección de la pantalla
    CMP A,0x032     ; Lo comparamos con el carácter ASCII de '2'
    JZ Print        ; Si es '2', saltamos a la subrutina 'Print'
    CMP A,0x031     ; Lo comparamos con el carácter ASCII de '1'
    JZ Borrar       ; Si es '1', saltamos a la subrutina 'Borrar'
    JMP             ; De otra forma, volvemos al loop del pulsador

Borrar:             ; Subrutina para borrar el tablero
    INC B           ; Incrementamos el puntero de los píxeles
    MOVB [B],0xFF   ; Movemos el color blanco a ese pixel
      CMP B,0x03FF  ; Verificamos si llegamos al último pixel
    JZ Pulsador     ; Si es así, volvemos al loop del pulsador
    JMP Borrar      ; Si no, volvemos a 'Borrar' para seguir borrando

Print:
    MOVB CL,0x00    ; Cargamos 0 en el registro CL
Bl:
    MOVB DL,0x00    ; Cargamos 0 en el registro DL
Blanco:
    INC B           ; Incrementamos el puntero de los píxeles
    CMP B,0x03FF    ; Verificamos si llegamos al último pixel.
```

```
    JZ Pulsador     ; Si es así, volvemos al loop del pulsador
    INCB CL         ; Si no, incrementamos CL
    INCB DL         ; Incrementamos DL
    CMPB CL,0x0002  ; Comparamos CL con '2'
     JZ Salto2      ; Si es así, vamos a 'Salto2'
    JMP Blanco      ; si no, a 'Blanco'
Ng:
    MOVB DL,0x00
Negro:
    MOVB [B],0x00
    INC B
    CMP B,0x03FF
    JZ Pulsador
    INCB CL
    INCB DL
    CMPB CL,0x0002
    JZ Salto1
    JMP Negro
Salto1:
    MOVB CL,0x00
    CMPB DL,0x0020
    JZ Ng
    JMP Blanco
Salto2:
    MOVB CL,0x00    ; Reiniciamos el contador C
    CMPB DL,0x0020  ; Comparamos D con 32
    JZ Bl           ; Si llegamos a 32, saltamos a 'Bl' para pintar con negro
    JMP Negro

    HLT
```

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 la pantalla.

c) Modifique el código de manera que en lugar de Mario Bross, 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

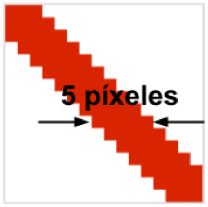
```
    JMP boot
vslDisplay EQU 0x300
sprite:
    DB "\xFF\xFF\xFF\xFF\xFF\xC4\xC4\xC4"
    DB "\xC4\xC4\xFF\xFF\xFF\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\xFF\xC4\xC4\xC4\xC4"
    DB "\xC4\xC4\xC4\xC4\xC4\xFF\xFF\xFF"
    DB "\xFF\xFF\xFF\xFF\x8C\x8C\x8C\xF4"
    DB "\xF4\x8C\xF4\xFF\xFF\xFF\xFF\xFF"
    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 regristro A el valor del color actual del valor
↪   de C
    CMP A, 0xC4         ; Valor de color verde, cuando el registro A tiene el color
↪   verde salta a chageColor
    JZ .changeColor
.paint:
    MOVB [D], AL        ; Coloca el pixel en la posicion de la pantalla actual.
    INC C
    INC D
    CMP D, 0x400
    JNZ .loop
    HLT
.changeColor:
    MOVB AL, 0x10
    JMP .paint
```

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.

Solución

```
;Cuadro De Ajedrez
  MOVB AL,0x0002
  MOVB BH,0x03
  MOVB CH,0x00
Secuencia2:
  MOVB CL,0x00
  MOVB [B],0x0C4
  CMPB BL,255
  JZ Fin
  INCB BL
  CMPB BL,240
  JZ Salto2
  INCB AL
  CMPB AL,0x0005
  JZ Secuencia1
  JMP Secuencia2
Secuencia1:
  MOVB AL,0x00
  INCB BL
  CMPB BL,0x000F
  JZ Salto1
  INCB CL
  CMPB CL,0x000C
  JZ Secuencia2
  JMP Secuencia1
Salto2:
  INCB BL
  JMP Secuencia1
Salto1:
  INCB BL
  INCB AL
  MOVB [B],0x00
  JMP Secuencia2
Fin:
  HLT
```

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.
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.
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.
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).
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 descriptas 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

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.