ARQUITECTURA DE COMPUTADORAS (E1213)
CIRCUITOS DIGITALES Y MICROPROCESADORES (E0213)
Curso 2022 - Versión 4

ACTIVIDADES PRÁCTICAS PARA EL MÓDULO 2
Microcontroladores y la arquitectura AVR

Entrega individual obligatoria en Moodle de los Problemas 9, 10 y Proyecto

Objetivos del Módulo

Introducción al repertorio de instrucciones de AVR. Comparación con MARIE. Primeros programas simples en assembler. Introducción al ATmega32P, el kit de desarrollo y Atmel Studio. Simulación y depurado de programas. Introducción a los periféricos e interrupciones. Ports de entrada/salida y timers.
Se recomienda repasar previamente la **Guía para la resolución de la Práctica 2**, disponible en Moodle.

Problema 1

A fin de hacer una primera comparación de la arquitectura AVR con MARIE, se propone implementar un programa simple que realice la suma de dos números enteros de 16 bits que se encuentran almacenados en la memoria SRAM.
Antes de comenzar descargue una versión actualizada del documento "AVR Instruction Set Manual". Se recomienda utilizar una versión pdf con índice para facilitar la exploración del documento.
Inspeccione la documentación ofrecida en el manual para las instrucciones de suma (ADD y ADC). Verifique las diferencias respecto de la modificación del Status Register (SREG). ¿De qué otras operaciones aritméticas dispone AVR?
Analice las instrucciones de carga y almacenamiento con direccionamiento directo (LDS y STS). Note que existen dos versiones de cada instrucción. ¿Cuál es la diferencia?
¿Qué otros modos de direccionamiento están disponibles para la suma y la carga?
Una vez comprendidos los detalles de funcionamiento de las instrucciones que va a utilizar, proponga el código para la suma de 16 bits en AVR y compare con el código equivalente en MARIE:

a) número de instrucciones necesarias,
b) cantidad de bytes de memoria de programa,
c) ciclos de reloj necesarios para completar la tarea.

Problema 2

Desarrolle un programa para AVR que realice el producto entre dos números positivos, donde el multiplicando y el producto sean de dos bytes y el multiplicador de un byte: a) Utilizando la instrucción MUL. b) Sin utilizar la instrucción MUL. Calcule el tiempo de ejecución de cada procedimiento. ¿Qué rango de validez tiene cada variable?, ¿Qué ocurre si se lo sobrepasa? ¿Cómo solucionaría este inconveniente? ¿La instrucción MUL está disponible en todas las implementaciones de AVR? Finalmente, proponga un algoritmo adecuado para AVR que realice el cociente entre dos números positivos, donde el divisor y el cociente sean de un byte y el dividendo de dos bytes.

Problema 3

Identifique los modos de direccionamiento disponibles en AVR, sus limitaciones, ventajas y desventajas de cada uno respecto de las diferentes zonas del mapa de memoria. ¿Qué opina respecto de la ortogonalidad del repertorio de instrucciones de AVR?

a) Realice un ensamblado manual del siguiente código, completando los datos faltantes en la tabla. Verifique utilizando Atmel Studio.

Dirección de Memoria	Instrucción en Hexadecimal	Código Mnemotécnico	Tipo de Direccionamiento	Registro de Estado ITHSVNZC
0x00000000		ldi r16,$0		
		mov r0,r16		
		ldi r16,$ff		
		mov r1,r16		
		ldi yh,$02		
		ldi y1,$01		
		ldi r16,$08		
		sts $200,r16		
		ldi r16,$03		
		sts $201,r16		
		lds r17,$202		
		mov r0,r17		
		movw r0,y		
		ld r0,y+		
		ld r0,-y		
		ldd r0,y+2		

b) ¿A qué secuencia de instrucciones corresponde la siguiente secuencia binaria almacenada en memoria de programa? Expresada en hexa:
40 E1 55 E1 64 2F 45 17 2C F4 43 95 00 00 00 00 0C 94 03 00 00 00 98 95
c) Proponga una secuencia de instrucciones AVR que invierta el orden de un arreglo de caracteres que se encuentra almacenado en los registros r0~r9 y almacene el arreglo invertido en los registros r10~r19.
d) Compare con el código que resulta si el arreglo de caracteres se encuentra almacenado en la memoria interna, comenzando en la dirección 0x0201 y cuya cantidad de elementos consecutivos se indica en la dirección 0x0200.

Problema 4
Reescriba para AVR el algoritmo del Problema 4 de la Práctica 1. ¿Cuáles son las instrucciones que facilitan el trabajo en este caso?. Compare la utilización de memoria de programa y de datos. Proponga una forma de comparar la performance de ambos procesadores. ¿A qué frecuencia de reloj debería operar MARIE para equiparar la performance de AVR operando a 16 MHz?

Problema 5
Diseñe y codifique para AVR una subrutina que retorne el mínimo de un vector. Verifique su funcionamiento en ATmel Studio. Recuerde inicializar correctamente la pila para poder utilizarla.

Problema 6
Genere una cadena de caracteres codificados en ASCII con su apellido en minúsculas seguido por un caracter nulo y su número de alumno. La cadena debe comenzar en la dirección 0x0100 y terminar con el caracter CR=0x0D (Carriage Return, retorno de carro). Luego, utilizando una subrutina, reemplace las letras minúsculas por mayúsculas y los caracteres 0 por caracteres Espacio (SP=0x20). Almacene la nueva cadena en la dirección 0x0120.

Problema 7
En dos direcciones consecutivas de memoria se encuentra contenido un BCD menor a 10000. Indique en cuatro posiciones de memoria consecutivas las cantidades de unidades de mil, centenas, decenas y unidades que contiene dicho número.

Problema 8
a) Proponga un programa que detecte errores de información dispuesta en codificación de paridad simple con 7 bits de datos y 1 de paridad. Estime el tiempo de retardo de la subrutina. Verifique el funcionamiento del detector de errores con los siguientes paquetes de datos codificados con el método de paridad: 11010111, 11010101, 10010101 (los bits están codificados como p7b6b5b4b3b2b1b0 en donde los 7 bits menos significativos de 1 byte son de datos y el más significativo es de paridad).
b) Implemente el problema anterior en el kit de manera que, cada vez que se presione un pulsador conectado a un pin del MCU, se decodifiquen los bytes propuestos. Si se detecta un error en la paridad, se debe encender el LED.
c) Implemente configurando el pin del pulsador como una interrupción externa. Indique cuál es la diferencia respecto a la implementación del inciso a).

Problema 9
Utilizando el LED disponible en el kit escriba un programa que lo prenda y apague 10 veces por segundo utilizando un retardo de 0.05 segundos con una rutina bloqueante (wasted cycle). ¿Qué desventaja presenta este método de temporización? ¿Cómo debería implementarse una rutina no-bloqueante? Implemente el inciso anterior de forma no bloqueante utilizando interrupciones y el módulo temporizador. ¿Cómo organizaría el programa para incorporar una tarea de background?

Problema 10
Un sistema tiene como objetivo detectar presencia mediante un sensor de barrera infrarroja (emular con un pulsador). Cuando el sistema detecta presencia debe emitir una señal intermitente de periodo 500ms por medio de una lámpara roja (emular con un led) durante 5 segundos. Implemente la detección del pulsador y la temporización mediante interrupciones. Teniendo en cuenta que el sistema funciona a batería, luego de realizar la secuencia de detección, el MCU debe entrar en modo bajo consumo hasta la próxima detección.

Proyecto, parte II: algoritmo de ordenamiento
Implemente en AVR la solución propuesta en la Práctica 1 para la subrutina de ordenamiento de un vector. Se propone en este caso trabajar con un vector de largo máximo de 256 elementos de 8 bits. Se recomienda almacenar el largo del vector como una constante utilizando la directiva .SET del ensamblador, para poder modificarlo luego. El vector deberá ubicarse en la memoria RAM, comenzando en la dirección 0x0100.

Para que no sea exactamente el mismo problema agreguemos un desafío adicional: implementar una subrutina que complete el vector con números aleatorios. Esto permitirá, con múltiples corridas, calcular el tiempo medio, máximo y mínimo que demora el algoritmo en ordenarlo.

Verifique el correcto funcionamiento en el simulador. Mida su performance y la utilización de memoria. Compare con la solución obtenida para MARIE.

La performance de su solución quedará caracterizada por los siguientes parámetros, los cuales deberán ser indicados claramente en el informe:
a) Tiempo medio de la iteración generación-ordenamiento del vector de 256

elementos (en ms).

b) Ocupación de la memoria de programa (en palabras de 16 bits).

c) Ocupación de la memoria de datos (en bytes). No incluir el vector.

Para realizar una comprobación experimental de los resultados obtenidos por simulación puede utilizarse en el kit uno de los puertos de salida visualizado en el osciloscopio. Puede implementarse, por ejemplo, la siguiente secuencia:

- Encender la salida
- Generar el vector aleatorio de largo máximo
- Apagar la salida
- Ordenar el vector
- Repetir

Una estrategia de este tipo debería proporcionar una indicación precisa de la performance de ambas rutinas en conjunto. Observe las fluctuaciones y calcule el tiempo medio, máximo y mínimo de la iteración.
Si no se dispone de un osciloscopio, puede utilizarse el LED disponible en el kit como indicación. Deberá utilizarse una estrategia adecuada, ya que el ojo humano no permite individualizar destellos de frecuencia mayor a 25 Hz.

Informe

El informe debe incluir los siguientes ítems:
a) Análisis comparativo de la solución obtenida para AVR, respecto de la solución para MARIE (variables y registros, modos de direccionamiento, iteraciones, etc.).
b) Simulación en Microchip Studio. Medición de tiempos, ocupación de memoria de datos y memoria de programa, como se indicó anteriormente.
c) Expresión y cálculo de la "mejora" del algoritmo de ordenamiento, respecto de un procesador MARIE operando a la misma frecuencia.
d) Verificación experimental, incluyendo una descripción del método utilizado. Los tiempos medidos deben coincidir (dentro del margen de error de la medida realizada) con los tiempos simulados en b).
e) Comparación con los resultados obtenidos por los compañeros.
f) Conclusiones.

Protocolo obligatorio de entrega para los problemas resueltos

a) Comprobar el correcto funcionamiento de cada una de las soluciones en el simulador y en el kit.

b) Incorporar a cada código un encabezado moderado, incluyendo nombre completo y número del alumno, práctica y problema, fecha, versión, etc.

c) Incorporar los comentarios que crea necesarios, sobre todo en los puntos clave del código. Recuerde que los comentarios deben ayudar al docente a comprender su solución, no hace falta explicar cómo funciona el procesador.

d) Nombrar el archivo solución de cada ejercicio como NNNNN_P_E.asm, donde NNNNN es el número de alumno, P es el número de práctica (en este caso 2) y E es el número de ejercicio (para el proyecto utilizar el número 0 cero). El proyecto debe acompañarse de un informe NNNNN_P_0.pdf donde se explica el método que se utilizó para medir la performance y la ocupación de memoria, junto con los resultados obtenidos.

e) Comprobar que el funcionamiento de los programas no se haya alterado debido a las modificaciones introducidas en los puntos b), c) y d).

f) Juntar todos los entregables de la misma práctica en un único archivo llamado NNNNN_P.zip y subirlo en el formulario a continuación. Verificar fecha límite de entrega para cada práctica.

No utilice directorios dentro del archivo comprimido.
No incluya archivos que no respeten el formato.

Ejemplo: Se considerará que el alumno número 54321/6 entregó la Práctica 2 (contiene 2 ejercicios obligatorios y el proyecto) si, dentro del plazo estipulado, subió el archivo 54321_2.zip, conteniendo los archivos:

54321_2.zip -> 54321_2_9.asm
54321_2_10.asm
54321_2_0.asm
54321_2_0.pdf

Si la entrega no respeta el formato se considerará como no presentada.

Evaluación

La entrega dentro del plazo estipulado, el formato correcto de la misma y el correcto funcionamiento de los programas, aseguran la aprobación con 6. La nota entre 6 y 10 se determinará considerando el grado de elaboración de las soluciones propuestas, su correcta documentación y el nivel de optimización alcanzado.