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

ACTIVIDADES PRÁCTICAS PARA EL MÓDULO 1
Arquitectura básica de procesadores y primer repertorio de instrucciones

Entrega individual obligatoria de todos los problemas y el proyecto

Objetivos del Módulo
Elaboración de los primeros programas en lenguaje ensamblador para la arquitectura didáctica MARIE. Estructuras de control y subrutinas. Vectores y punteros. Introducción a la simulación y evaluación de algoritmos.
Se recomienda revisar previamente la Guía para el Módulo 1, disponible en Moodle, la cual contiene algunos problemas resueltos y un resumen de las principales características de la Arquitectura MARIE.

Problema 1
Proponga un programa para MARIE que calcule, utilizando una iteración, los primeros 10 múltiplos de 7 y los almacene ordenados en forma creciente en la memoria, comenzando en la dirección 0x0050. Compare la iteración resultante con con las estructuras de alto nivel conocidas (for, if, while, etc.). Compare con el código que resultaría si en lugar de utilizar una iteración, se utilizara sólo una secuencia de instrucciones.

Problema 2
Escriba un programa que calcule el largo de una cadena de caracteres que comienza en la dirección 0x0100 y termina con el carácter CR= 0x0D (Carriage Return, retorno de carro).

Problema 3
Escriba un programa para MARIE que determine la cantidad de números negativos en un bloque de datos. La longitud del bloque se encuentra almacenada en la dirección de memoria 0x0010 y el bloque comienza en la dirección 0x0011. Envíe el resultado a la salida.

Problema 4
Escriba un programa que calcule los primeros 20 elementos de la sucesión de Fibonacci y los almacene en orden creciente en posiciones consecutivas de memoria empezando en 0x0040. Verificación: El vigésimo término debe valer 0x1055 y debe quedar almacenado en la posición de memoria 0x53.

Problema 5
Implemente como subrutina el programa de multiplicación iterativa sin signo que se ofrece como ejemplo en el simulador y se discute en la guía de ejercicios resueltos. Utilice dos posiciones de memoria fijas para pasar los argumentos y una más para el resultado.

Problema 6
Escriba una subrutina que calcule el factorial de un número entero utilizando la subrutina de multiplicación del problema anterior. ¿Puede explotarse la recursividad de la operación factorial? Recuerde que: n! = n*(n-1)!

Proyecto Integrador: Algoritmos de ordenamiento

Proponga un programa que ordene en forma creciente un vector de números enteros de 16 bits en Ca2. La longitud del vector se encuentra almacenada en la posición de memoria 0x001 y el vector comienza en la dirección 0x002. El resultado del programa debe ser almacenado en la misma ubicación del vector original.

Formato del programa

A fin de uniformar las entregas se proponen el siguiente esqueleto y encabezado para la solución:

```
/ Algoritmo de ordenamiento en MARIE
/ Nombre completo y número de alumno
/ Algoritmo utilizado: ...
/ Ocupación de memoria de programa: ...
/ Ocupación de memoria de datos (sin incluir el vector): ...
/ Performance
/ Vector ordenado    -3,-2,-1,0,1,2,3 : ... instrucciones
/ Vector invertido   3,2,1,0,-1,-2,-3 : ... instrucciones
/ Vector desordenado 0,-2,1,3,-1,2,-3 : ... instrucciones

ORG 000
Jump ordenar          / Vector de reset

VECTOR, DEC 7         / Largo del vector
DEC -3                / Primer elemento del vector
DEC -2
DEC -1
DEC 0
DEC 1
DEC 2
DEC 3

P1, DEC 0             / Variables y constantes
P2, DEC 0             / Nombrar adecuadamente y describir su utilización
...

ordenar, ...          / INSTRUCCIONES
...
```

Para realizar una evaluación del correcto funcionamiento y su performance, se propone un primer test con los tres vectores indicados en el encabezado. Se solicita realizar las tres mediciones sobre su algoritmo e incluir en el encabezado los resultados. Utilizaremos el número de instrucciones ejecutadas como indicador de performance, suponiendo que todas las instrucciones toman la misma cantidad de ciclos de reloj.

Para la entrega, al programa en assembler (.ASM) debe adjuntarse un informe (.PDF) describiendo la estrategia utilizada para resolver el problema, los resultados de la verificación funcional y las mediciones adicionales de performance que considere necesarias para caracterizar correctamente la solución.

Guía para la resolución del proyecto

Antes de empezar a programar, recomendamos considerar los siguientes puntos: ¿Qué algoritmo de ordenamiento va a utilizar? ¿Cuáles son las posibilidades? ¿Cuál de esas posibilidades se adapta mejor a los recursos disponibles en MARIE? ¿Qué vector usará para verificar su funcionamiento? ¿Existen vectores más (o menos) desordenados? ¿El grado de desorden depende del algoritmo utilizado para

ordenarlo?

El ordenamiento de vectores es un problema que admite múltiples soluciones y que ha sido estudiado exhaustivamente en disciplinas matemáticas e informáticas. Su interés radica en el impacto indirecto que tiene en la performance de algoritmos más complicados. Puede intentar tomar dimensión del problema en https://en.wikipedia.org/wiki/Sorting_algorithm. Cuanto mejor conozca el problema, más eficiente será su solución.

Una vez elegido el algoritmo, éste debe ser implementado con los escasos recursos disponibles en MARIE. Seguramente le será útil realizar un diagrama de flujo antes de empezar a programar. Esto le permitirá administrar eficientemente la arquitectura.

Tenga en cuenta que el código resultante será un poco más largo que el de los problemas anteriores. Por lo tanto es importante redactar código ordenado, bien comentado, con sus variables y constantes nombradas adecuadamente.

Una vez obtenida una solución, el siguiente paso será verificar que la operación de la misma sea correcta. Esto significa que el programa siempre debe ordenar el vector en orden numéricamente creciente. Algunas preguntas que hay que hacerse: ¿La implementación realizada tiene algún límite? Si es así, los límites deben documentarse. ¿Cómo puede hacer para asegurar que ordenará bien todos los vectores posibles? ¿Qué vectores se utilizarán para la verificación? También debe documentarse brevemente la verificación funcional que se realizó.

Cumplido el objetivo de la verificación funcional, surge una cuestión un poco más complicada: ¿Cuán "buena" es la solución obtenida y cómo se compara con otras posibles soluciones? Para eso vamos a definir dos criterios diferentes, los cuales podrían ser antagónicos:

a) ¿Cuán rápido ordena el vector? (performance)
Por el momento supondremos que todas las instrucciones de MARIE demoran el mismo tiempo en ejecutarse. Por lo tanto, podemos inferir que la mejor performance será la del programa que ejecute menos instrucciones para lograr su objetivo. Es importante notar que no se trata del número de instrucciones que tiene el código del programa, sino de cuántas instrucciones se ejecutan realmente para realizar el ordenamiento. Tenga en cuenta que una repetición se ejecuta múltiples veces, y que una selección ejecuta sólo una de las dos opciones. Además, tenga en cuenta que el tiempo de ejecución dependerá del largo del vector, por supuesto. Pero también, dependiendo del algoritmo seleccionado, podría depender del contenido del vector. Sería útil verificar cuánto tarda el programa en completarse si le damos un vector ordenado, o uno en orden inverso, o un vector con todos los elementos nulos y otro con todos los elementos iguales distintos de cero.

b) ¿Qué cantidad mínima de memoria necesita el programa para funcionar? (recursos)
En principio puede diferenciarse entre memoria de programa y memoria de datos, aunque en MARIE se trata del mismo recurso. Respecto de la memoria de programa, todas las instrucciones de MARIE ocupan 16 bits (una posición de la memoria). Respecto de la memoria de datos, deberá tenerse en cuenta que para resolver el problema se deberán utilizar posiciones de memoria adicionales para almacenar variables temporales, punteros, constantes, etc. El programa más compacto será el que requiera de menos posiciones de memoria total para almacenar datos e instrucciones, sin importar cuánto tarda el programa en ejecutarse.

Dada la cantidad de alternativas habrá tantas soluciones como alumnos y serán muy diferentes entre sí. Para compararlas y evaluarlas utilizaremos los siguientes parámetros:

- Verificación funcional.
- Correcta documentación.
- Cantidad de instrucciones que ejecuta el programa para ordenar un vector de un largo determinado.
- Ocupación de memoria de programa y de datos (sin contar el vector).

Recomendamos intercambiar estos números con sus compañeros antes de entregar, para verificar que su solución se encuentre bien orientada.

Protocolo obligatorio de entrega para los problemas resueltos

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

b) Incorporar al 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 MARIE.

*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 1) 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 algoritmo empleado, los detalles de su implementación, el método que se utilizó para caracterizar la performance y 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 en Moodle. Verificar fecha límite de entrega.

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 1 (la cual contiene 6 ejercicios y el proyecto) si, dentro del plazo estipulado, subió en Moodle el archivo 54321_1.zip, conteniendo los archivos:

```
54321_1.zip  ->  54321_1_1.asm
                 54321_1_2.asm
                 54321_1_3.asm
                 54321_1_4.asm
                 54321_1_5.asm
                 54321_1_6.asm
                 54321_1_0.asm
                 54321_1_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.