DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (II/2019)

Tarea 2: Arreglos, Saltos y Subrutinas

Fecha límite de entrega: Miércoles 25 de Septiembre a las 21:00

Motivación

En esta tarea practicarán, usando el assembly del computador básico visto en clases, cómo usar arreglos, saltos y llamar subrutinas en un lenguaje de bajo nivel. Entendiendo así la ejecución de algoritmos conocidos en un computador.

Objetivos

1. Practicar el uso del assembly del computador básico.[1]
2. Practicar el uso de saltos en assembly.
3. Practicar la escritura de subrutinas en assembly.
4. Practicar el manejo de arreglos en assembly.
5. Investigar sobre un algoritmo e implementarlo en assembly.

Introducción

Mientras el cuerpo de ayudantes de Arquitectura de Computadores sumaba los altísimos puntajes obtenidos por los alumnos en la Tarea 1, se quemó la calculadora de tanto computar. Por suerte encontraron un computador primigenio de 8 bits, capaz de correr programas escritos en el Assembly del curso, en la abandonada oficina del temido profesor Hans. Pero, hay un problema: los puntajes son tan altos que se van a necesitar nuevos programas para adaptar su funcionamiento y así lograr el objetivo de entregar las notas a tiempo. Sin embargo, al encender el aparato y cargar los puntajes, un distraído ayudante introdujo una galleta en el lector de Diskette, y creemos que esto generó un CookieBug,[2] provocando que los datos se desordenen. Se te ha pedido a ti que, luego de sacar el polvo a este viejo computador, escribas los programas requeridos en el Assembly del curso que resuelva esta galletosa situación.

[1] Pueden revisar la ISA y microarquitectura en detalle en la sección 5.5 de los Apuntes del curso.
[2] mejor conocido como "Harvard Mark II bugs"

Comentarios generales

Deberás realizar tres programas en el assembly del computador básico, cuya arquitectura es Harvard, la cual trabaja con no más de 8 bits y complemento a dos.

Todos sus programas deberán ejecutarse en el emulador que se encuentra en el **Syllabus**. Para cada uno se te proporcionará un archivo `.txt` con un código base como el que se presenta a continuación:

```
DATA:

//////////////////Variables Corrección/////////////////////////////////////
                                                                        //
  nombre1         valor1        //                                      //
                                                                        //
  nombre2         valor2        //                                      //
                                                                        //
  arreglo         dato0         //                                      //
                  dato1         //                                      //
                  dato2         //                                      //
                  dato3         //                                      //
                                                                        //
  nombre3         valor3        //                                      //
                                                                        //
////////////////////////////////////////////////////////////////////////

//! begin [inicio código alumno]

//////////////////Variables Alumno//////////////////////////////////////


      //Definir Variables Aquí


//////////////////Fín Variables Alumno//////////////////////////////////

CODE:

  JMP main

nombre_label:

//////////////////Código Alumnos//////////////////////////////////////


      // Completar Código Aquí


//////////////////Fín Código Alumnos//////////////////////////////////

//! end [fin código alumno]

//////////////////Corrección/////////////////////////////////////////////
                                                                        //
end: JMP end                    // Fín                                  //
                                                                        //
main:                                                                   //
                                                                        //
  JMP nombre_label              //                                      //
                                                                        //
  NOP                           // Reservado para Corrección            //
  NOP                           //                                      //
  NOP                           //                                      //
  NOP                           //                                      //
                                                                        //
  JMP end                       // Terminar                             //
                                                                        //
////////////////////////////////////////////////////////////////////////
```

Todo lo que esté entre los dos comentarios `//! begin` y `//! end` es tuyo (con las restricciones que se encuentran a continuación), lo puedes editar como quieras agregando variables en DATA y código en CODE, **salvo la instrucción de salto a main y la label definida a continuación.** Nosotros reemplazaremos el contenido que esté fuera de ambos comentarios especiales al momento de corregir.

Considera que la cantidad de instrucciones `NOP` que aparezcan forma parte del código, no puedes eliminarlos en caso de que tengas problemas de espacio. Recuerda también que no puedes asumir que los espacios libres tienen guardado el valor 0 inicialmente, el emulador dará una advertencia en caso de leer memoria no inicializada.

Sobre la corrección y calificación

La corrección será nuevamente automática y el puntaje se asignará por medio de *tests* focalizados. Por esta razón es muy importante que respetes la estructura del código que se te indicó.

Entrega

El periodo de entrega finaliza el Miércoles 25 de Septiembre a las 21:00 hrs. Debes entregar cada parte de tu tarea por medio del cuestionario correspondiente en **SIDING** , cada programa debe ir como un archivo `.zip` llamado `T2_parte_X_YYYYYYYY.zip`. Donde **X** es el número de la pregunta e **YYYYYYYY** tu número de alumno. Cada archivo `.zip` a entregar debe de contener los siguientes archivos:

- Un archivo `.txt` del problema correspondiente, con nombre `T2_parte_X_YYYYYYYY.txt`, donde `X` es el número de la pregunta, e **YYYYYYYY** tu número de alumno. Debes entregarlo aunque no realices el problema.
- Un archivo `README.md` con tu nombre completo, número de alumno y una explicación del algoritmo que se pretendió implementar y su equivalente en pseudocódigo. Todos los archivos auxiliares para tu *Readme* (como imágenes, música, vídeos, etc.), deberán estar en una carpeta `assets` en caso de haberlos.

No seguir el formato de entrega anteriormente detallado significará que la tarea no fue entregada en el plazo correspondiente.[3] Tampoco se aceptarán entregas por un medio que no sea el especificado, salvo por motivos de fuerza mayor.

[3]Dependiendo de la gravedad del incumplimiento se aceptará re-corrección aplicando descuentos.

1. ¿En la T1 el promedio fue 1,00045? (1,5 puntos)

Problema propuesto

Deberás realizar un programa que sea capaz de sumar dos números enteros de 16 bits, los que estarán en complemento a 2. Finalmente debes guardar el resultado en memoria, en una ubicación diferente según la magnitud del resultado.

El programa debe de tener una estructura de ejecución secuencial, realizando lo pedido al saltar al label declarado al inicio de la sección CODE, tal como se indica en el anexo. Esto no impide que utilicen instrucciones de salto dentro del código.

Secciones DATA y CODE entregadas

Se te entregarán en la sección DATA, 4 valores, los cuales corresponden a los bytes de ambos números que debes sumar. Considera que estarán almacenados en formato **big endian**.

Formato valores de salida

Deberás guardar el puntaje en distintos lugares dentro de la memoria según si es mayor o menor a cero:

- Si el resultado es mayor o igual a 0 debes guardarlo en las variables res_a y res_b.
- Si el resultado es menor a 0, deberás guardarlo en las variables res_c y res_d.
- El resultado debe guardarse siguiendo la lógica de *big endian*.

En el anexo se te mostrará el formato del **.txt** que se te entregará para esta pregunta.

2. Sacarse un 7 está de moda (2 puntos)

Problema propuesto

Deberás recorrer un arreglo ordenado de largo variable, cuyos datos corresponden a notas en una extraña escala, calcular la moda del mismo,[4] para luego guardar el resultado en memoria según se te indique. En caso de que el arreglo sea bimodal, deben elegir la moda que esté más lejos del inicio del arreglo, es decir la que comience en una dirección de memoria más alta.

El arreglo estará ordenado, de manera que todos los datos que sean iguales estarán juntos.[5] En esta pregunta deberás realizar un programa que se ejecute de forma secuencial realizando lo pedido al saltar al label declarado al inicio de la sección CODE. Esto no impide que utilices instrucciones de salto dentro del código.

Secciones DATA y CODE entregadas

Se te entregarán en la sección DATA 3 variables. La primera corresponde al arreglo de datos, y la segunda a la cantidad de datos del arreglo (el largo del arreglo). Finalmente, la variable resultado es para guardar la moda resultante. Puedes agregar otras variables auxiliares al final de la sección DATA si así lo estimas conveniente.

Formato valores de salida

Tal y como se menciona anteriormente, al finalizar, debes guardar la moda en la dirección `resultado`.

En el anexo se te mostrará el formato del **.txt** que se te entregará para esta pregunta.

[4] Valor con más frecuencia en una distribución de datos.

[5] Por ejemplo 1 1 2 2 3 3 3 ...

3. AssemblySort (2,5 puntos)

Problema propuesto

Dado un conjunto de arreglos y sus largos variables, deberás implementar un algoritmo capaz de ordenarlo.

Deberás realizar una subrutina que pueda ser llamada con la label declarada al inicio de la sección CODE. Esto no impide que utilicen instrucciones de salto y otras subrutinas dentro del código.

Puedes asumir que el largo máximo de cada arreglo no superará los 20 elementos y que cada elemento tendrá un valor entre -60 y 60 incluidos.

Algunos de los algoritmos que puedes implementar son:

- Selection Sort
- Insertion Sort
- Bubble Sort

Puedes implementar algún otro, pero aquí se mencionan los más simples.

Secciones DATA y CODE entregadas

En la sección DATA irán como variables los arreglos y su largos, y al final de la sección podrán agregar variables auxiliares si así lo estiman conveniente.

Formato valores de salida

Deberás guardar el arreglo ordenado en la misma dirección de memoria en la que estaba el arreglo original.

En el anexo se te mostrará el formato del **.txt** que se te entregará para esta pregunta.

Contacto

Cualquier pregunta sobre la tarea, ya sean de enunciado, contenido o sobre aspectos administrativos deben comunicarse con los ayudantes creando issues en el Syllabus del Github del curso.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno (grupo) copia un trabajo, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir reprobación del curso y un procedimiento sumario. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.

4. Anexo

Código base parte 1

```
DATA:

/////////////////Variables Corrección//////////////////////////////////////
                                                                        //
  num_1_1         120   // Primer byte del primer número                //
  num_1_2         34    // Segundo byte del primer número               //
                                                                        //
  num_2_1         0     // Primer byte del segundo número               //
  num_2_2         255   // Segundo byte del segundo número              //
                                                                        //
  res_a           0     // Parte del resultado si es mayor o igual a 0  //
  res_b           0     // Parte del resultado si es mayor o igual a 0  //
                                                                        //
  res_c           0     // Parte del resultado si es menor a 0          //
  res_d           0     // Parte del resultado si es menor a 0          //
                                                                        //
////////////////////////////////////////////////////////////////////////

//! begin [inicio código alumno]

/////////////////Variables Alumno/////////////////////////////////////////


      //Definir Variables Aquí


/////////////////Fín Variables Alumno//////////////////////////////////////

CODE:

  JMP main

special_adder:

/////////////////Código Alumnos///////////////////////////////////////////


      // Completar Código Aquí


/////////////////Fín Código Alumnos//////////////////////////////////////

//! end [fin código alumno]

/////////////////Corrección///////////////////////////////////////////////
                                                                        //
end: JMP end                    // Fín                                  //
                                                                        //
main:                                                                   //
                                                                        //
  JMP special_adder             // Súmar                                //
                                                                        //
  NOP                           // Reservado para Corrección            //
  NOP                           //                                      //
  NOP                           //                                      //
  NOP                           //                                      //
                                                                        //
  JMP end                       // Terminar                             //
                                                                        //
////////////////////////////////////////////////////////////////////////
```

Código base parte 2

```
DATA:

//////////////////Variables Corrección//////////////////////////////////////
                                                                          //
  arreglo         -35           //  Arreglo                             //
                  -35           //                                      //
                  -30           //                                      //
                  -29           //                                      //
                  -28           //                                      //
                  -26           //                                      //
                  -16           //                                      //
                  -5            //                                      //
                  15            //                                      //
                  23            //                                      //
                  26            //                                      //
                  26            //                                      //
                  28            //                                      //
                  31            //                                      //
                  31            //                                      //
                  35            //                                      //
                  40            //                                      //
                  44            //                                      //
                  51            //                                      //
                  55            //                                      //
                                                                        //
  largo_arreglo   20            // Largo del Arreglo                    //
                                                                        //
  resultado       0             // Aquí deben guardar el resultado      //
                                                                        //
////////////////////////////////////////////////////////////////////////////

//! begin [inicio código alumno]

//////////////////Variables Alumno//////////////////////////////////////////


      //Definir Variables Aquí


//////////////////Fín Variables Alumno//////////////////////////////////////

CODE:

  JMP main

sacar_moda:

//////////////////Código Alumnos//////////////////////////////////////////


      // Completar Código Aquí


//////////////////Fín Código Alumnos//////////////////////////////////////

//! end [fin código alumno]

//////////////////Corrección//////////////////////////////////////////////////
                                                                          //
end: JMP end                    // Fín                                  //
                                                                        //
main:                                                                   //
                                                                        //
  JMP sacar_moda                //                                     //
                                                                        //
  NOP                           // Reservado para Corrección            //
  NOP                           //                                      //
```

```
  NOP                       //                                  //
  NOP                       //                                  //
                                                                //
  JMP end                   // Terminar                         //
                                                                //
////////////////////////////////////////////////////////////////////
```

Código base parte 3

```
DATA:

////////////////Variables Corrección//////////////////////////////////
                                                                //
  arreglo_1       5         //  Arreglo 1                       //
                  3         //                                  //
                  -3        //                                  //
                  2         //                                  //
                  -7        //                                  //
                  18        //                                  //
                  7         //                                  //
                  -1        //                                  //
                  20        //                                  //
                  -30       //                                  //
                  20        //                                  //
                  -33       //                                  //
                  55        //                                  //
                  -19       //                                  //
                  60        //                                  //
                  -18       //                                  //
                  53        //                                  //
                  12        //                                  //
                  5         //                                  //
                                                                //
  largo_arreglo_1 19        // Largo del Arreglo 1              //
                                                                //
  arreglo_2       52        // Arreglo 2                        //
                  35        //                                  //
                  -53       //                                  //
                  22        //                                  //
                  -3        //                                  //
                  23        //                                  //
                  12        //                                  //
                  57        //                                  //
                  21        //                                  //
                  42        //                                  //
                  -12       //                                  //
                  -35       //                                  //
                  14        //                                  //
                  -23       //                                  //
                  23        //                                  //
                  32        //                                  //
                                                                //
  largo_arreglo_2 16        // Largo del Arreglo 2              //
                                                                //
////////////////////////////////////////////////////////////////////

//! begin [inicio código alumno]

////////////////Variables Alumno/////////////////////////////////////


     //Definir Variables Aquí


////////////////Fín Variables Alumno/////////////////////////////////

CODE:

```

```
  JMP main                    // Iniciar Programa

//////////////////Subrutinas Alumnos/////////////////////////////////////


      // Definir Subrutinas Aquí


ordenar_arreglo:              // Recibe Largo en A, Puntero en B

      // Completar Código Aquí

  RET


//////////////////Fín Subrutinas Alumnos//////////////////////////////////

//! end [fin código alumno]

//////////////////Corrección/////////////////////////////////////////////////
                                                                          //
end: JMP end                  // Fín                                      //
                                                                          //
main:                         // Llamadas para ordenar Arreglos           //
                                                                          //
  MOV A,(largo_arreglo_1)     // Largo del Arreglo 1 en A                 //
  MOV B,arreglo_1             // Puntero al Arreglo 1 en B                //
  CALL ordenar_arreglo           // Ordenar Arreglo                       //
                                                                          //
  MOV A,(largo_arreglo_2)     // Largo del Arreglo 2 en A                 //
  MOV B,arreglo_2             // Puntero al Arreglo 2 en B                //
  CALL ordenar_arreglo        // Ordenar Arreglo                          //
                                                                          //
  NOP                         // Reservado para Corrección                //
  NOP                         //                                          //
  NOP                         //                                          //
  NOP                         //                                          //
  NOP                         //                                          //
  NOP                         //                                          //
  NOP                         //                                          //
  NOP                         //                                          //
  NOP                         //                                          //
                                                                          //
  JMP end                     // Terminar                                 //
                                                                          //
//////////////////////////////////////////////////////////////////////////////
```