Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (I/2018)

Enunciado Tarea 01

Programación Recursiva en Assembly

Fecha de entrega: 13 de Abril del 2018, 23:59 hrs.

1. Descripción

Para esta tarea, vamos a programar. La diferencia con otros ramos es que ahora será en un lenguaje de bajo nivel: Assembly. Para esta tarea van a tener que generar dos programas en Assembly que serán descritos a continuación.

2. Primer Algoritmo: seudo-palíndromo (40 % de la nota)

Un palíndromo son aquellas palabras o expresiones que son iguales si se leen de izquierda a derecha y de derecha a izquierda. Un seudo-palíndromo es aquella palabra o expresión que solo requiere cambiar una letra para que ahora sea palíndromo. Ejemplo:

Palíndromo	Seudo-palíndromo
Atar a la rata.	Atar a la rate.
Ana lava lana.	Ana lava gana.

Para esta tarea deberán generar un programa que, dada una palabra o expresión seudo-palíndromo, diga la posición de la letra que es necesario cambiar.

Inputs y Output

Para este programa el input serán 3 variables:

1. **letra:** Esta variable deberá indicar la posición de la letra. Por defecto la variable parte como 0.
2. **largo:** Esta variable indicará el largo de la palabra o expresión. Pueden suponer que siempre será mayor a 0.
3. **palindromo:** Esta variable tendrá la palabra o expresión seudo-palíndromo.

En Assembly este *input* se verá literalmente así:

```
1 DATA:
2 //HEAD
3 letra 0
4 largo 13
5 palindromo "ANA_LAVA_LANE"
6 //HEAD
7  otras_variables....
```

Observaciones importantes

1. Las posiciones de las letras parten desde 0, es decir, la posición 0 es la primera letra, la posición 1 es la segunda, etc.

2. **No puede asumir** que todos los casos serán seudo-palíndromo, es decir, se pueden ingresar palabras que tengan 2 o más letras que sea necesario cambiar para formar el palíndromo. En ese caso, el resultado debe ser `FFFF`

3. **Deben** indicar la letra con menor posición a cambiar. En el ejemplo "ANA LAVA LANE", se puede cambiar la "E" de la posición 12 por "A" o cambiar la "A" de la posición 0 por "E". La respuesta correcta es cambiar la letra de la posición 0.

4. Los espacios no son considerados letras. Por lo tanto, deben omitirlos en el análisis.

5. Puede asumir que solo usaremos caracteres de la tabla ASCII y estos solo serán letras en mayúscula y el espacio. Puede revisar la tabla en esta página: https://ascii.cl/es/.

6. Puede crear toda la variable que quiera pero siempre debajo de la expresión, es decir, donde dice "otras_variables...".

7. Los espacios cuentan en las posiciones. En el ejemplo de "ANA LAVA LANE", la posición 4 es un espacio.

El *output* será editando la variable "letra" en donde deberán poner la respuesta en dicha variable, es decir, hacer "`MOV (letra), resultado`". En el ejemplo de "ANA LAVA LANE", al final tendrían que hacer "`MOV (letra), 0`".

3. Segundo Algoritmo: Camino más largo (60 % de la nota)

Luego de trabajar con *strings*, daremos un avance para trabajar con matrices[1]. Para este algoritmo deberán implementar lo necesario para indicar la cantidad de nodos del camino más largo de un grafo acíclico.

Se define grafo acíclico como aquel que no posee ningún ciclo, es decir, no existe un camino donde partes de un nodo y llegas al mismo.

Para poder trabajar con grafos, utilizaremos su representación matricial. En particular, el grafo será codificado utilizando su matriz de adyacencia (revisar este link: https://es.wikipedia.org/wiki/Matriz_de_adyacencia), que será almacenada en orden de filas y comenzará en la dirección de memoria asociada al *label* "matriz". La variable "columnas" indica el número de columnas de la matriz. El resultado del algoritmo (la cantidad de nodos del camino más largo) debe indicarse en la dirección asociada al label "camino".

```
1  DATA:
2  //HEAD
3  camino 0
4  columnas: 3
5  matriz 0
6          0
7          1
8          0
9          0
10         0
11         0
12         1
13         0
14 //HEAD
15  otras_variables....
```

Para ayudar con la visualización, la matriz del ejemplo se vería así:

```
0 0 1
0 0 0
0 1 0
```

Para ayudar, la siguiente imagen muestra un grafo junto con su matriz de adyacencia.

Interpretación:

1. El nodo 1 está conectado con el nodo 3.
2. El nodo 3 está conectado con el nodo 2

Observaciones importantes

- Un nodo jamás estará conectado con sí mismo.
- El grafo es dirigido: `https://es.wikipedia.org/wiki/Grafo_dirigido`

[1] Lo que no es tan fácil como es en Python.

- La matriz siempre es cuadrada.
- Puede crear toda la variable que quiera pero siempre debajo de la lista, es decir, donde dice "otras_variables..."

4. Corrección

Para corregir sus tareas los ayudantes usaremos un script que correrá su algoritmo escrito en Assembly y evaluará su output. Para esto usaremos un intérprete en el que hemos estado trabajando (escrito en Python, por lo que no deberían tener problemas con él). Los algoritmos serán recolectados y testeados de forma automática, por lo que es imprescindible de que sigan al pie de la letra las siguientes instrucciones:

- Es importante que entreguen en su repositorio asignado dos archivos con los nombres `p1.txt` y `p2.txt`, cada uno correspondiente a su algoritmo para resolver cada problema.
- Es importante que **NO BORREN NI MODIFIQUEN** los comentarios que dicen `//HEAD` ya que estos le indican al script las variables que debe reemplazar.
- El intérprete en el que estamos trabajando tiene instruccones especiales. Estas son `//PRT A` `//PRT B` y `//PRT (dir)`. Si escriben este comentario solo en una linea de su programa, el intérprete imprimirá en consola el valor del registro A, B o la dirección dir, según lo solicitado. Esto es importante debido a que para cada problema deberán imprimir sus resultados usando esta instrucción.
- Finalmente es importante resaltar que su programa en caso de no terminar, será evaluado como incorrecto, por lo que lo ideal es que su programa termine con una label `end:` y una vez que quieran terminar su programa hagan `JMP end` sin que éste quede en un loop final (es decir, no pueden hacer `JMP end` después de la label `end:`

Los algoritmos serán evaluados en base a diferentes tests. Cada test tiene asociado un puntaje binario, o sea "logrado" o "no logrado". En caso de haber cualquier problema con el intérprete (les rogamos paciencia, ya que es posible que existan algunos problemas) también tienen como herramienta el Assembler del curso (también pueden usarlo como una guía para sus programas).
Nota: Este Assembler solo funciona en el Sistema Operativo Windows.

5. Contacto

- Francesca Lucchini: flucchini@uc.cl
- Felipe Pezoa: fipezoa@uc.cl
- Hernán Valdivieso: hfvaldivieso@uc.cl
- Luis Leiva: lileiva@uc.cl

6. Integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios 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. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por

la Dirección de Docencia de la Escuela de Ingeniería.

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 para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, 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 copia un trabajo, obtendrá nota final 1,1 en el curso y se solicitará a la Dirección de Docencia de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.