ESTRUCTURA DE COMPUTADORES

GRADO EN INGENIERÍA INFORMÁTICA

uc3m | Universidad Carlos III de Madrid

Grupo de Arquitectura de Computadores

Práctica 1
Introducción al lenguaje ensamblador

Curso 2017/2018

Contenido

Objetivos de la práctica

El objetivo de la práctica consiste en entender los conceptos básicos relacionados con la programación en ensamblador. Para ello, se utilizará como ejemplo el ensamblador del MIPS32 y el simulador QtSpim. La práctica consta de 2 ejercicios.

Ejercicio 1

El objetivo de este ejercicio es desarrollar un programa en ensamblador que procese un fichero de entrada formado por palabras, que podrán estar separadas por uno o más blancos o saltos de línea. El objetivo del programa es determinar el número de palabras del texto que incluyen como subcadena otra dada.

El segmento de datos del programa incluirá, al menos, las siguientes definiciones:

```
.data:

    nombre_fichero: .asciiz "c:\users\fichero.txt "
    cadena: .asciiz "na"
```

Donde `nombre_fichero` será el nombre del fichero a procesar. El programa deberá indicar cuántas palabras de las incluidas en el fichero de entrada tienen a la cadena `na` como subcadena. Por ejemplo, considere el fichero de entrada:

```
Este fichero es un ejemplo de prueba para
Determinar cuantas palabras del mismo tienen na como subcadena
```

Para este fichero de prueba, el programa debería imprimir el valor 3. En caso de que el fichero no pueda ser abierto (el fichero no existe o la apertura del mismo falla), el programa deberá imprimir el siguiente mensaje de error:

```
Error al abrir el fichero
```

Si el fichero está vacío, el programa deberá imprimir en la consola “Fichero Vacío”.

El programa deberá, además, hacer todo el tratamiento de errores necesario. Tenga en cuenta que el programa debe poder ejecutar con cualquier fichero de texto y cualquier subcadena.

En el apartado del enunciado ***Procesamiento de ficheros con QtSpim***, se describe cómo procesar archivos con el simulador QtSpim.

Ejercicio 2

El objetivo de este ejercicio es desarrollar un programa en ensamblador que procese un fichero de entrada formado por palabras, que podrán estar separadas por uno o más blancos o saltos de línea. El programa deberá contabilizar la longitud de todas las palabras incluidas en el fichero y generará un fichero de salida en el que indicará el número de ocurrencias que aparecen en el texto de una determinada longitud y una línea final con la longitud media de las palabras leídas.

El segmento de datos del programa incluirá, al menos, las siguientes definiciones:

```
.data:

      ficheroEntrada: .asciiz "c:\users\ficheroE.txt"
      ficheroSalida: .asciiz "c:\users\ficheroS.txt"
```

Donde `ficheroEntrada` será el nombre del fichero de entrada a procesar. Por ejemplo, para el fichero de entrada:

```
Este fichero es un ejemplo de prueba para
El segundo ejercicio
```

Para este fichero de prueba, el programa generará el siguiente fichero de salida:

```
Longitud 2: 4
Longitud 4: 2
Longitud 6: 1
Longitud 7: 3
Longitud 9: 1

Longitud media:   4.727
```

Se mostrará una línea por cada longitud que realmente exista en el fichero, una línea en blanco y otra línea con la longitud media. Considere que la longitud máxima de la palabra a procesar es de 256 bytes.

En caso de que el fichero de entrada no pueda ser abierto (el fichero no existe o la apertura del mismo falla), el programa deberá imprimir el siguiente mensaje de error:

```
Error al abrir el fichero de entrada
```

En caso de que el fichero de salida no pueda ser creado, el programa deberá imprimir el siguiente mensaje de error:

```
Error al crear el fichero de salida
```

En caso de que el fichero de entrada esté vacío, el programa deberá imprimir el siguiente mensaje:

```
Fichero Vacío
```

El programa deberá, además, hacer todo el tratamiento de errores necesario. La longitud media deberá imprimirse con decimales en caso de ser necesario.

En el apartado del enunciado ***Procesamiento de ficheros con QtSpim***, se describe cómo procesar archivos con el simulador QtSpim.

Procesamiento de ficheros con QtSpim

El simulador QtSpim incluye una serie de llamadas al sistema para poder procesar ficheros, a continuación, se indican estas llamadas al sistema:

- Abrir un fichero que existe para solo lectura:
  - $a0: dirección de memoria a partir de la cual se almacena el nombre del fichero.
  - $a1: 0x000 (solo lectura).
  - $v0: 13 (código de la llamada al sistema)
  - Devuelve en $v0 el descriptor de fichero o -1 en caso de error.
- Abrir un fichero que existe para solo lectura:
  - $a0: dirección de memoria a partir de la cual se almacena el nombre del fichero.
  - $a1: 0x001 (solo escritura).
  - $v0: 13 (código de la llamada al sistema)
  - Devuelve en $v0 el descriptor de fichero o -1 en caso de error.
- Abrir un fichero que existe para lectura y escritura:
  - $a0: dirección de memoria a partir de la cual se almacena el nombre del fichero.
  - $a1: 0x002 (lectura-escritura).
  - $v0: 13 (código de la llamada al sistema.
  - Devuelve en $v0 el descriptor de fichero o -1 en caso de error.
- Crear un fichero (o truncarlo en caso de que exista) para escritura:
  - $a0: dirección de memoria a partir de la cual se almacena el nombre del fichero.
  - $a1: 0x601 (creación, truncado y solo escritura).
  - $a2: permisos del fichero (estilo UNIX), por ejemplo: 0x1FF (lectura y escritura).
  - $v0: 13 (código de la llamada al sistema).
  - Devuelve en $v0 el descriptor de fichero o -1 en caso de error.
- Cierre de un fichero
  - $a0: descriptor del fichero abierto.
  - $v0: 16 (código de la llamada al sistema).
- Lectura de un fichero:
  - $a0: descriptor de fichero.
  - $a1: dirección de comienzo del buffer donde almacenar los bytes leídos.
  - $a2: número de bytes a leer.
  - La llamada devuelve en $v0 el número de bytes realmente leídos, -1 en caso de error y 0 en caso de fin de fichero.
- Escritura en un fichero:
  - $a0: descriptor de fichero.
  - $a1: dirección de comienzo del buffer donde se almacenan los bytes a escribir.
  - $a2: número de bytes a escribir en el fichero.
  - La llamada devuelve en $v0 el número de bytes realmente escritos o -1 en caso de error.

El siguiente fragmento de código permite abrir un fichero para solo lectura y leer los dos primeros bytes del mismo. A continuación, crea un fichero de salida (truncándolo en caso de que exista y copia los dos caracteres leídos en él.

```
.data
      ficheroEntrada:   .asciiz "c:\tmp\ficheroE.txt"
      ficheroSalida:    .asciiz "c:\tmp\ficheroS.txt"
      buf: .space 256

.text
.globl main

main:
```

```
#abre el fichero de entrada
la    $a0, ficheroEntrada
li    $a1, 0x0
li    $v0, 13
syscall

#copia en $t0 el descriptor de fichero
move  $t0, $v0

#lee dos bytes
move  $a0, $t0,
la    $a1, buf
li    $a2, 2
li    $v0, 14
syscall

#cierra el fichero de entrada
move  $a0, $t0
li    $v0, 16
syscall

#crea el fichero de salida
la    $a0, ficheroSalida
li    $a1, 0x101
li    $a2, 0x1FF
li     $v0, 13
syscall


#copia en $t0 el descriptor de fichero
move  $t0, $v0

#copia los bytes leídos anteriormente
move  $a0, $t0
la    $a1, buf
li    $a2, 2
li    $v0, 15
syscall


#cierra el fichero de salida
move  $a0, $t0
li    $v0, 16
syscall

jr $ra
```

Tenga en cuenta que en el fragmento de código anterior no se ha realizado ningún tratamiento de errores. En los programas a desarrollar para esta práctica, sí deberá hacerse.

Aspectos importantes a tener en cuenta

Normas generales

1) La entrega de la práctica se realizará a través de los entregadores habilitados. No se permite la entrega a través de correo electrónico.
2) La entrega se realizará en el plazo dado por los entregadores. Es posible que para un entregador de Aula Global el fin del plazo para una entrega a las 24:00 termine 10 minutos antes. Revise el soporte de Aula Global.
3) Se prestará especial atención a detectar funcionalidades copiadas entre dos prácticas. En caso de encontrar implementaciones comunes en dos prácticas (o contenidos similares en la memoria), ambas obtendrán una calificación de 0.

Códigos de la práctica

1) Los dos ejercicios realizados han de hacer uso **obligatoriamente** de funciones. Para ello debe seguirse el convenio de paso de parámetros descrito en clase. Aquellos ejercicios que no incluyan ninguna función, aparte del main, serán calificadas con un 0.
2) Todos los ejercicios deben seguir el formato de salida que se pide en cada enunciado, incluyendo las líneas en blanco y mayúsculas mostradas en los ejemplos. Debe seguirse el formato de fichero definido en el enunciado.
3) Todo el código correspondiente al ejercicio 1 debe implementarse en una función denominada **ejercicio1**. Esta función recibirá dos argumentos: el primero será la dirección de inicio de la cadena que contiene el nombre del fichero y el segundo será la dirección de inicio de la cadena que incluye la subcadena. La función no devuelve ningún resultado. El segundo ejercicio se realizará en la función **ejercicio2** y acepta dos argumentos: la dirección de inicio de la cadena que almacena el nombre del fichero de entrada y como segundo argumento la dirección de inicio de la cadena que almacena el nombre del fichero de salida. La función no devuelve ningún resultado.
4) Han de entregarse los dos ejercicios solicitados.
5) Todos los ejercicios deben seguir el formato de salida que se pide en cada enunciado. Debe seguirse el formato de fichero definido en el enunciado.
6) Los ejercicios que no compilen o que no se ajusten a la funcionalidad y requisitos planteados, obtendrán una calificación de 0.
7) Un programa no comentado, obtendrá una calificación de 0.

Memoria de la práctica

1) La memoria (un único documento) tendrá que contener al menos los siguientes apartados:
    - Portada donde figuren los autores (incluyendo nombre completo, NIA y dirección de correo electrónico).
    - Índice de contenidos.

- Contenidos pedidos en los distintos ejercicios (una sección por ejercicio).
- Conclusiones y problemas encontrados.

2) **La longitud de la memoria no deberá superar las 10 páginas** (portada e índice incluidos).
3) Al respecto de la posible descripción de los programas pedidos:
    - Se ha de detallar las principales funciones implementadas. La memoria debe describir el comportamiento de los programas, así como las principales decisiones de diseño (adicionalmente se pueden incluir diagramas de flujo, algoritmos, etc.).
    - Se ha de incluir la batería de pruebas utilizadas y resultados obtenidos. Se dará mayor puntuación a pruebas avanzadas, casos extremos, y en general a aquellas pruebas que garanticen el correcto funcionamiento de la práctica en todos los casos.
        - Evite pruebas duplicadas que evalúan los mismos flujos de programa. La puntuación de este apartado no se mide en función del número de pruebas, sino del grado de cobertura de las mismas. Es mejor pocas pruebas que evalúan diferentes casos a muchas pruebas que evalúan siempre el mismo caso.

<u>NOTA:</u> NO DESCUIDE LA CALIDAD DE LA MEMORIA DE SU PRÁCTICA.

Aprobar la memoria es tan imprescindible para aprobar la práctica, como el correcto funcionamiento de la misma. Si al evaluarse la memoria de su práctica, se considera que no alcanza el mínimo admisible, su práctica estará suspensa.

Procedimiento de entrega de la práctica

La entrega de la práctica 1 se realizará de forma electrónica a través de Aula Global

La fecha límite de entrega para ambos es el día **7 de noviembre de 2017 a las 23:55 horas.**

Es posible entregar tantas veces como quiera dentro del plazo dado, la única versión registrada de su práctica es la última entregada. La valoración de la práctica es la valoración del contenido de esta última entrega. Revise siempre lo que entregue.

Entregador: Se deberá entregar un único archivo comprimido en formato **zip** con el nombre ec_p1_AAAAAAAAA_BBBBBBBBB.zip donde A…A y B…B son los NIA de los integrantes del grupo.

El archivo **zip** debe contener solo los siguientes archivos:

- **ejercicio1.s**
- **ejercicio2.s**
- **memoria.pdf**

Evaluación de la práctica

La evaluación de la práctica se va a dividir en dos partes:

- **Código (7 puntos)**
- **Memoria (3 puntos)**

La puntuación de cada ejercicio será:

- Ejercicio 1 (5 *puntos*)
- Ejercicio 2 (5 *puntos*)

Si un ejercicio no se entrega su puntuación será 0. Tenga en cuenta que para seguir el proceso de evaluación continua, la nota mínima obtenida en cada práctica debe ser de 2 y la media de las dos prácticas 4.

NOTAS:

1. **Si se detecta un error de concepto grave en la práctica (en cualquier apartado de cualquier ejercicio), la valoración global de toda la práctica será de cero puntos (0 puntos).**
2. **En caso de encontrar implementaciones comunes en dos prácticas (o contenidos similares en la memoria), ambas obtendrán una calificación de 0.**
3. **En caso de encontrarse fragmentos de código obtenidos directamente de Internet, la práctica tendrá una calificación de 0.**