Universidad Simón Bolívar.
Departamento de Computación
CI-3815 Organización del Computador.
Sep-Dic 2014.

Proyecto # 2 (17 %)

El objetivo es diseñar, codificar, ensamblar y ejecutar programas escritos en lenguaje ensamblador del procesador MIPS.

1 Descripción

Como problema de aplicación de la programación en ensamblador del procesador MIPS elegimos una técnica de codificación de datos conocida como LZW (LempelZivWelch). Este tipo de técnicas se usan ampliamente para aprovechar de mejor manera la capacidad de los dispositivos de almacenamiento y para reducir los tiempos de transferencia de datos, al reducir el tamaño de la información que se desea almacenar o transferir.
En muchas aplicaciones, la fuente de datos posee patrones recurrentes. Considere como ejemplo cualquier libro de texto en el cual ciertos patrones o palabras se repiten una y otra vez. Una solución muy razonable para codificar este tipo de fuentes de datos consiste en mantener un diccionario con los patrones que ocurren con más frecuencia. Cuando estos patrones aparecen en la fuente de datos, se codifican mediante una referencia al diccionario. Si un patrón en la fuente de datos no está en el diccionario, se codifica empleando algún otro método.

2 Lempel-Ziv-Welch (LZW)

Para describir el algoritmo LZW supondremos que se aplica sobre cadenas de caracteres (como puede ser un texto). Básicamente, el algoritmo construye durante su ejecución un diccionario de cadenas de caracteres, el cual es inicializado con una entrada por cada símbolo del alfabeto de la fuente de datos.

A medida que el algoritmo evoluciona, agrega nuevas cadenas de caracteres al diccionario de manera que una cadena se agrega solamente si el prefijo con un caracter menos ya está en el diccionario. Por ejemplo, la cadena de caracteres Juan se agrega al diccionario sólo si la cadena Jua apareció previamente en la secuencia de entrada. Veamos cómo funciona el algoritmo mediante un ejemplo. Consideremos la siguiente secuencia de entrada:

ABCABCA

Asumiendo que el alfabeto de la fuente de datos es A, B, C, el diccionario inicialmente tiene la Tabla 1. El codificador comienza por la letra A de la

Indice	Entrada
0	A
1	B
2	C

Table 1: Diccionario Inicial

secuencia de entrada. Esta entrada está en el diccionario, de manera que la concatena con la siguiente letra, formando la cadena AB. Esta cadena no está en el diccionario, así que codifica la letra A con el índice que tiene asociado en el diccionario, (en nuestro caso 0) y agrega la cadena AB al diccionario como la cuarta entrada. Ver Tabla 2

El algoritmo continúa con la cadena que comienza con la letra B. Como

Indice	Entrada
0	A
1	B
2	C
3	AB

Table 2:

B está en el diccionario, la concatena con la siguiente letra para formar la

cadena BC. Esta cadena no está en el diccionario, por lo que codifica la letra B con su índice en el diccionario, que es 1 y agrega la cadena BC al diccionario como la quinta entrada. Ver Tabla 3, y continua con la cadena a partir de

Indice	Entrada
0	A
1	B
2	C
3	AB
4	BC

Table 3:

la letra C. Como C está en el diccionario, la concatena con la siguiente letra para formar la cadena CA. Esta cadena no está en el diccionario, por lo que codifica la letra C con su índice en el diccionario, que es 2 y agrega la cadena CA al diccionario como la sexta entrada. Ver Tabla 4, y comienza a formar

Indice	Entrada
0	A
1	B
2	C
3	AB
4	BC
5	CA

Table 4:

una nueva cadena con la letra A. La siguiente letra de la secuencia de entrada es B, formando la cadena AB. Esta cadena ya existe en el diccionario (índice 3), de manera que se lee la siguiente letra, que es C, para formar la cadena ABC. Esta cadena no existe en el diccionario, por lo que la cadena AB se codifica con su índice del diccionario, que es 3 y se agrega al diccionario como la séptima entrada la cadena ABC. Ver Tabla 5, y se comienza una nueva

Indice	Entrada
0	A
1	B
2	C
3	AB
4	BC
5	CA
6	ABC

Table 5:

cadena con la letra C. La siguiente, y última letra de la secuencia de entrada es A, formando la cadena CA. Esta cadena ya existe en el diccionario, así que se codifica con su índice del diccionario, que es 5. El diccionario al final del proceso de codificación luce como se muestra en la Tabla 6. La salida producida por el algoritmo LZW es la secuencia 0 1 2 3 5. Si la misma secuencia de entrada se hubiera codificado caracter por caracter hubieran sido necesarios 7 bytes. Por lo tanto, el algoritmo LZW alcanza un factor de compresin 1.4:1 en este ejemplo (asumiendo que cada índice del diccionario se codifica en un byte).

Indice	Entrada
0	A
1	B
2	C
3	AB
4	BC
5	CA
6	ABC

Table 6: Diccionario Final

A continuación se presenta el algoritmo LZW.

```
# el simbolo + que aparece en este algoritmo es el operador concatenar
 LZW ( )
 string s;
 char c;
{
    Inicializar_diccionario;
    s = string_vacio;
    Mientras ( hay datos que leer)
    {
        c = leer_caracter;
        Si ( s+c es un string del diccionario )
           {
                s = s+c
           }
        sino
           {
                imprimir_codigo de s
                agregar s+c al diccionario
                s = c
            }
    }
}
```

3 Detalles de implementación

- Para la implementación del algoritmo LZW deben diseñar una estructura de datos para almacenar el diccionario.
- El alfabeto está compuesto por todas las letras mayúsculas (A, B, C, ..., Z).
- Realizar una programación basada en el uso de funciones de manera que la estructura modular del programa facilite su comprensión y favorezca la reutilización de código cumpliendo con la convención de responsabilidad compartida vista en clase.

- Su implementación debe solicitar el texto a ser codificado y dar como resultado la secuencia de índices.
- Su código debe de estar comentado.

4 Material Apoyo

- http://en.wikipedia.org/wiki/Lempel-Ziv-Welch
- Nelson, Mark. LZW Data Compression. Dr. Dobb's Journal, Oct 1989
- Ziv,J. and A.Lampel,"A Universal Algorithm foe Sequential Data Compression",IEEE Transactions on Information Theory,vol 23,no 3,1977

Fecha de entrega: Viernes 07-11-2014. Debe, en horas de clase, entregar el documento impreso donde explica y justifica sus decisiones de diseño, y el día anterior hacer submit del archivo a Moodle.