Universidad Rey Juan Carlos

Arquitectura de Computadores/Arquitectura de Sistemas Audiovisuales II

Práctica 1: Codificación de datos e instrucciones

Katia Leal Algara

INTRODUCCIÓN:

El primer paso consiste en comprobar cómo un programa fuente en ensamblador del MIPS32 se carga en memoria y cómo el código binario resultante, que se muestra en hexadecimal, se muestra en el segmento de datos y de código. En esta sesión de prácticas se fijarán conceptos relacionados con el repertorio de instrucciones, con la codificación de instrucciones y con la ordenación de los bytes en memoria.

A. Segmento de texto

Cargar en el simulador el programa de prueba “Hola Mundo”:

```
      .data
str:  .asciiz "Hola Mundo en MIPS32\n"
      .text
main: la $a0, str
      li $v0, 4
      syscall
      li $v0, 10
      syscall
```

En el segmento de texto, analiza el almacenamiento en memoria de las instrucciones. Contesta a las siguientes preguntas:

1. ¿Cuál es la longitud de las instrucciones?
2. ¿Cuál es la dirección de la segunda instrucción?
3. ¿Qué tipo de ordenación se utiliza? ¿Cómo lo sabes?

Consulta el manual del juego de instrucciones y analiza la representación y el almacenamiento en hexadecimal de las instrucciones.

4. Teniendo en cuenta los campos de cada una de las instrucciones, ¿es correcta la codificación en hexadecimal?
5. Clasifica cada una de las instrucciones según su tipo, I, R o J.
6. Indicar el modo de direccionamiento empleado por cada una de las instrucciones.

B. Segmento de datos

Cargar en el simulador el programa de prueba “Hola Mundo”. Analiza el almacenamiento de los datos en el segmento de datos. Contesta a las siguientes preguntas:

1. ¿Cuál es la longitud de una dirección de memoria?
2. ¿Cuál es la dirección del dato *str*?
3. ¿Qué tipo de ordenación se utiliza? ¿Cómo lo sabes?
4. ¿Cuántos bytes emplea la variable *str*? ¿cuántas *palabras* ocupa en memoria?

C. Más tipos de datos

```
.data
.asciiz "Suma de 2 y 2 = %d\n"
.byte 10,11,12,13,14,15,16,17,18,19,0
.text
```

1. ¿Cuántos bytes emplea la variable de tipo .asciiz? ¿cuántas palabras ocupa en memoria?
2. ¿Cuántos bytes emplea la variable de tipo .byte? ¿a qué estructura de datos te recuerda?

D. Almacenamiento de bytes

¿Cómo se almacenarían en memoria los siguientes datos?

```
.data
.asciiz "Suma de 2 y 2 = %d\n"
.byte 10
.byte 11
.byte 12
.byte 13
.byte 14
.byte 15
.byte 16
.byte 17
.byte 18
.byte 19
.byte 0
```

```
      .text
```

Piensa primero cómo crees que quedaría la memoria y después comprueba el resultado en el simulador.

E. Arrays y carga de memoria a registro

Escribe el siguiente código en un fichero:

```
      .data
array:  .word 10,11,12,13
        .byte 0x1a,0x0b,10
        .ascii   "Simulador MARS"
        .asciiz  ", MIPS32"
index:  .word 1
        .text
main:   lw      $t0, array($zero)
        lw      $t5, index($zero)
        addi    $t4, $zero, 4
        mult    $t4, $t5
        mflo    $t6
        lw      $t1, array($t6)
        addi    $t5, $t5, 1
        mult    $t4, $t5
        mflo    $t6
        lw      $t2, array($t6)
        addi    $t5, $t5, 1
        mult    $t4, $t5
        mflo    $t6
        lw      $t3, array($t6)
        li      $v0, 10
        syscall
```

Ejecuta el programa paso a paso. Fíjate en la ventana de registros y observa los valores que van tomando algunos de ellos. Para comprender esos valores, observar detenidamente las operaciones descritas por las instrucciones del código. Estudia cómo se cargan en los registros las distintas posiciones del array de datos en memoria, y qué operaciones aritméticas es necesario realizar (y porqué). Escribe un breve resumen sobre el estudio realizado.