Sesión 9- Computador Virtual COVI

Arquitectura de Computadores I (IS19)

Área de Arquitectura y Tecnología de Computadores
Departamento de Ingeniería y Ciencia de los Computadores

Curso 2002-2003

1. Objetivos

En esta práctica se realiza la ejecución de un programa en ensamblador DLX sobre el Computador Virtual (COVI). Posteriormente, se evalúan distintos parámetros de ejecución (CPI, tiempo de ejecución y MIPS) para distintas configuraciones de la ruta de datos. El objetivo de esta práctica es la aplicación de las diferentes métricas vistas en clase en la evaluación de determinados benchmarks y la comparación entre diferentes máquinas para esos mismos programas de prueba.

2. Descripción del material a emplear

Para el desarrollo de esta sesión de laboratorio se dispondrá de un computador personal PC, con el sistema operativo Ms Windows y el Computador Virtual (COVI) instalado. Si esto no fuese así se podrá recurrir a la página web de la asignatura IS19:

http://www.unoweb-s.uji.es/IS19/1049205157/index_html

donde también estará disponible este guión y el archivo fuente sumav.s que se utiliza.

3. Desarrollo de la sesión de prácticas

3.1 Entorno de simulación y organización subDLX

(a) Cargar el código fuente ensamblador `sumav.s`. Elegir unidad de control cableada, y ruta de datos de ciclo largo. Abrir la ventana "Organización subDLX", identificar los bloques combinacionales y secuenciales que componen el computador.

(b) Comprobar en la ventana "Código Fuente" que el programa cargado se corresponde con el de la práctica.

(c) Abrir la ventana "Memoria de Instrucciones". ¿A partir de qué dirección de memoria se ha ubicado el código? Identificar las columnas con el código máquina y el código ensamblador.

(d) Comprobar que el segmento de datos se ha cargado correctamente (ventana "Memoria de Datos"). ¿Qué codificación y representación de los valores emplea mCOVI?.

(e) Abrir la ventana: "R.R.D", que muestra los registros de la ruta de datos. Localizar dichos registros en la organización subDLX. Abrir la ventana "Banco de Registros".

(f) Ejecutar el código ciclo a ciclo (F5). Observar las líneas activas y la evolución de los valores de los registros de la ruta de datos. Repítelo las veces que sea necesario hasta comprender el esquema de ejecución del programa.

(g) Repetir los puntos anteriores cargando el programa sumav.s con la configuración de ruta de datos de ciclo corto.

3.2 Medida de prestaciones

(a) Ejecutar completamente el programa `sumav.s`. Consultar las estadísticas y calcular el tiempo de ejecución en los dos procesadores mCOVI (ruta de datos ciclo largo y corto). Suponer las siguientes frecuencias de reloj:

- Fciclo_largo = 1,4 GHz
- Fciclo_corto = 2 GHz

Recuerda:

$$Tej = NI \times CPI \times Tc$$

siendo:

- I : número de instrucciones ejecutadas.
- CPI : número medio de ciclos por instrucción
- Tc : tiempo de ciclo.

(b) Calcular los MIPS (millones de instrucciones por segundo) que ejecutarían ambos procesadores.

(c) Para comparar dos computadores ejecutando el mismo código ensamblador, qué es mejor, ¿comparar tiempos de ejecución o MIPS?

(d) Confeccionar una tabla resumen de Tej y CPI para los dos procesadores (ruta de datos ciclo largo y corto).

4. Anexo: Listado de `sumav.s`

```
;;*******************************************************
;;* sumav.s
;;* Suma los elementos de un vector
;;* Arquitectura de Computadores (Ingeniería Industrial)
;;* Autor: Jesús Alastruey
;;* ATC - DIIS - Universidad de Zaragoza
;;*******************************************************

      .data 100           ;; Inicio del segmento de datos

      ; Constantes
      N:         .word 32        ;; Longitud del vector (en bytes)
      TAM:       .word 4         ;; Tamaño de un elemento del vector
                                 ;; (en bytes)

      ; Variables
      vector: .word 1, 4, -5, -6, 24, -8, -5, 16
      resul:  .space 4
```

```
        .text 200           ;; Inicio del segmento de instrucciones
        .global _main
_main:  lw      r1,N        ;; r1 = longitud del vector (N bytes)
        add     r2,r0,r0    ;; r2 = índice recorrido vector
        lw      r3,TAM      ;; r3 = tamaño un elemento (TAM bytes)
        add     r4,r0,r0    ;; r4 = acumulador suma elementos
bucle:  slt     r6,r2,r1    ;; Si r2<r1 -> r6=1, cc r6=0
        beqz    r6, fin     ;; Si r6=0 salta a fin
        lw      r5,vector(r2) ;; r5 = contenido de @(vector + r2)
        add     r4,r4,r5    ;; Sumo elemento actual
        add     r2,r2,r3    ;; r2 = r2 + 4
        bnez r6, bucle      ;; Salta siempre a bucle
fin:    sw      resul,r4    ;; Almacena resultado
        trap 0              ;; Final del programa
```