Sesión 8- Práctica de Medida de Rendimiento

Arquitectura de Computadores I (IS19)
Arquitectura de Computadores I (II19)

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

Curso 2002-2003

1. Objetivos

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 Linux, y una serie de benchmarks, que se encuentran en la página web de la asignatura II19 (http://icc2.act.uji.es/ii19/laboratorio.html). En concreto se emplearán los siguientes:

- Linpack: Programa que resuelve un sistema de ecuaciones 100x100. El resultado se presenta en KFLOPS.
- Dhrystone: Programa sintético para la evaluación del sistema mediante programación con enteros. Incluye las versiones 1.1 y 1.2 de dicho benchmark.
- Whetstone: Programa sintético para la evaluación del sistema mediante programación con números en coma flotante.
- gcc: Compilador del lenguaje de programación C, que sólo utiliza aritmética entera.

3. Desarrollo de la sesión de prácticas

3.1- Análisis de las prestaciones de las arquitecturas.

En este apartado se van a medir las prestaciones de los computadores del laboratorio en el que se realizan las prácticas empleando los benchmarks citados en el apartado anterior.

a) En primer lugar se ejecutarán los benchmarks *linpack, Dhrystone y Whetstone,* para ver su funcionamiento y qué tipo de resultados presentan. A continuación se describe el proceso de compilación y ejecución para cada uno de ellos:

- Linpack:
  - Compilación: *make linkpackc*
  - Ejecución: *lpc_dpr*

- Dhrystone:
  - Compilación: *make dhry21*
  - Ejecución: *dhry21* indicando que se van a realizar un millón de iteraciones.

- Whetstone:
  - Compilación: *gcc whteston.c –lm –o whet*
  - Ejecución: *whet*

Este proceso se debe repetir para otras dos máquinas diferentes, y extraer conclusiones sobre qué máquina presenta mejor comportamiento.

b) A continuación se ejecutarán los cuatro benchmarks presentados, midiendo los tiempos de ejecución con la ayuda de la orden *time*. Dicha orden presenta el tiempo de real de ejecución, el tiempo de usuario y el tiempo del sistema, medidos en segundos.
El proceso de ejecución para cada uno de ellos es el siguiente:

- Linpack: Se debe ejecutar *time linkpackc*
- Dhyrstone: Se debe ejecutar *time dhry21*
- Whetstone: Se debe ejecutar *time whes*
- gcc: Se debe ejecutar *time gcc whetston.c –lm–o whet*

Se deben apuntar los resultados obtenidos (tiempo de usuario en segundos) para poder compararlos con otras máquinas. Este proceso se debe repetir para otras dos máquinas diferentes, con el objetivo de rellenar la siguiente tabla que mostrará los resultados obtenidos tras ejecutar dichos benchmarks en las tres máquinas:

Programa/Máquina	A	B	C
Linpack			
Dhyrstone			
Whetstone			
gcc			

Comparar las prestaciones de los tres computadores empleando tres formas distintas. La primera considerará cada uno de los programas de un modo aislado, la segunda empleará la media aritmética de los tiempos de ejecución, y la tercera la media geométrica de los tiempos de ejecución normalizados a una de las tres máquinas.

3.2- Cálculo de los CPI.

En este apartado se va a obtener el valor del CPI para el siguiente programa escrito el lenguaje C (ejemplo.c), empleando la expresión T_{ejec}=*I x CPI x tiempo_ciclo_reloj*:

```
int main(int argc, char *argv[])
{

 int i,final;
 double a1,b1,a2,b2,a3,b3,a4,b4,r1,r2,r3,r4;

 a1=2.0;
 b1=2.5;
 a2=3.14;
 b2=56.9;
 a3=223.3;
 b3=-232.555555;
 a4=12.021212121;
 b4=-202929.212125;
 final=atoi(argv[1]);
 final=final*10000;
 for(i=0;i<=final;i++){
  r1=a1-b1;
  r2=a2+b2;
  r3=a3*b3;
  r4=a4/b4;
 }
}
```

A continuación se muestra el código generado en lenguaje ensamblador de la familia x86, obtenido tras ejecutar la orden *$gcc –S ejemplo.c:*

```
.file  "ejemplo.c"
       .version     "01.01"
gcc2_compiled.:
.section     .rodata
       .align 8
.LC0:
       .long 0x0,0x40000000
       .align 8
.LC1:
       .long 0x0,0x40040000
       .align 8
.LC2:
       .long 0x51eb851f,0x40091eb8
       .align 8
.LC3:
       .long 0x33333333,0x404c7333
       .align 8
.LC4:
       .long 0x9999999a,0x406be999
       .align 8
.LC5:
       .long 0x1b478423,0xc06d11c7
       .align 8
.LC6:
       .long 0x50abf295,0x40280adc
       .align 8
.LC7:
       .long 0xb26e978d,0xc108c589
```

```
.text
	.align 16
.globl main
	.type	main,@function
main:
	pushl %ebp
	movl %esp,%ebp
	subl $120,%esp
	fldl .LC0
	fstpl -16(%ebp)
	fldl .LC1
	fstpl -24(%ebp)
	fldl .LC2
	fstpl -32(%ebp)
	fldl .LC3
	fstpl -40(%ebp)
	fldl .LC4
	fstpl -48(%ebp)
	fldl .LC5
	fstpl -56(%ebp)
	fldl .LC6
	fstpl -64(%ebp)
	fldl .LC7
	fstpl -72(%ebp)
	addl $-12,%esp
	movl 12(%ebp),%eax
	addl $4,%eax
	movl (%eax),%edx
	pushl %edx
	call atoi
	addl $16,%esp
	movl %eax,%eax
	movl %eax,-8(%ebp)
	movl -8(%ebp),%edx
	movl %edx,%ecx
	movl %ecx,%eax
	sall $5,%eax
	subl %edx,%eax
	sall $2,%eax
	addl %edx,%eax
	leal 0(,%eax,4),%edx
	addl %edx,%eax
	movl %eax,%edx
	sall $4,%edx
	movl %edx,-8(%ebp)
	movl $0,-4(%ebp)
	.p2align 4,,7
.L3:
	movl -4(%ebp),%eax
	cmpl -8(%ebp),%eax
	jle .L6
	jmp .L4
	.p2align 4,,7
.L6:
	fldl -16(%ebp)
	fsubl -24(%ebp)
	fstpl -80(%ebp)
	fldl -32(%ebp)
	faddl -40(%ebp)
	fstpl -88(%ebp)
	fldl -48(%ebp)
	fmull -56(%ebp)
	fstpl -96(%ebp)
```

```
        fldl -64(%ebp)
        fdivl -72(%ebp)
        fstpl -104(%ebp)
.L5:
        incl -4(%ebp)
        jmp .L3
        .p2align 4,,7
.L4:
.L2:
        movl %ebp,%esp
        popl %ebp
        ret
.Lfe1:
        .size   main,.Lfe1-main
        .ident "GCC: (GNU) 2.95.2 19991024 (release)"
```

En primer lugar, se compilará y se ejecutará

$gcc ejemplo.c –o ejemplo
$time ejemplo 1000

Se anota el tiempo de ejecución del usuario. Seguidamente, se estimarán las instrucciones que el procesador ha ejecutado, a partir del código en ensamblador mostrado anteriormente. Como se puede observar el programa comienza a partir de la etiqueta *main*. En la etiqueta *.L3* comienza el bucle. El cuerpo del bucle tiene 18 instrucciones. El final del bucle está en la etiqueta *.L4*. Por tanto, el número de instrucciones ejecutadas, despreciando las instrucciones externas al bucle, es *I=18N*, siendo *N* el número de iteraciones del bucle (1000 en la ejecución realizada).
Se puede conocer la frecuencia de reloj del procesador mediante *$more /proc/cpuinfo*. A partir de este dato ya se puede obtener el *Tiempo_ciclo_reloj* y el valor de CPI.
A partir de estos valores se puede calcular el valor de los MIPS obtenidos por este programa, así como los MFLOPS, considerando que el programa de alto nivel realiza cuatro operaciones en coma flotante por cada iteración del bucle.

4. Evaluación

Se deberá entregar una memoria con los resultados obtenidos, convenientemente justificados y comentados. Para entregar dicha memoria se establecerá una entrevista concertada entre el profesor de prácticas y el alumno durante la semana del 5 al 9 de Mayo de 2003.