Procesadores Comerciales (3º Grado Informática)

Arq**T**ec**C**om - Dpto. Informática e Ingeniería de Sistemas.

CPS – UniZar

Práctica 4: Simulador de un predictor de saltos

Los objetivos de esta práctica son: i) comprender el funcionamiento de un predictor de saltos en un procesador OOO, ii) modelar varios predictores de saltos (local, global, hibrido), iii) analizar el comportamiento de los predictores y cuantificar su influencia en las prestaciones del procesador.

1. Descripción del simulador

En esta práctica se define y posteriormente se implementa un predictor de saltos. Lee detenidamente este apartado y estudia el código que te proporcionamos. El predictor consta de dos estructuras:

- Branch Target Buffer (BTB). Predice dirección destino de salto.
- Branch History Table (BHT). Predice si el salto es tomado o no tomado (sentido).

En un procesador real, la información proporcionada por el predictor de saltos se usa para guiar a la etapa de búsqueda de instrucciones durante los siguientes ciclos. Cuando la instrucción de salto llega a la etapa de ejecución, calcula la dirección destino y evalúa la condición, y los compara con la predicción. Si la predicción fue correcta la instrucción de salto no tiene efecto. Si la predicción no fue correcta se provoca una recuperación que implica eliminar todas las instrucciones posteriores al salto en el ROB y en todas las etapas del segmentado, e insertar el PC correcto en la etapa de búsqueda.

Nuestro generador de traza no es capaz de recorrer un camino incorrecto. Sólo nos proporciona traza del camino correcto. Sin embargo, ya en la etapa de búsqueda podemos consultar toda la información necesaria para determinar si el salto se predice correctamente o no. Para simular el comportamiento temporal de los saltos, tras un salto que se predice correctamente, la etapa de búsqueda continua por el camino proporcionado por el generador de traza sin ninguna penalización. Tras un salto cuya predicción falla, bloquearemos la etapa de búsqueda hasta que dicho salto llegue a la etapa de ejecución.

Los ficheros de partida para esta práctica se encuentran en el directorio de pilgor:

```
/export/home/alumnos/a01/saltos
```

Definiremos cuatro funciones para consultar y actualizar las tablas BTB y BHT. Se ha modificado el fichero cpu.c incluyendo el código que llama a estas funciones, contabiliza e imprime fallos del predictor y modela las paradas necesarias en caso de fallo de predicción de salto.

Encontraréis las cabeceras de las cuatro funciones en el fichero saltos.c. Describimos a continuación estas funciones:

```
char leerBTB (unsigned long long pc, unsigned long long *destino)
```

Recibe el PC de una instrucción. Si BTB determina que el PC corresponde a una instrucción de salto devuelve *cierto* y coloca en *destino la dirección predicha para el destino de salto. Devuelve *falso* en caso contrario.

```
char leerBHT (unsigned long long pc)
```

Recibe el PC de una instrucción. Devuelve *cierto* si BHT predice *tomado* para este salto. Devuelve *falso* en caso contrario. La información de PC sólo se usa en alguno de los modelos de predicción.

```
void actualizarBTB (unsigned long long pc, unsigned long long destino)
```

Recibe el PC de una instrucción de salto y la dirección destino de salto calculada por dicha instrucción. Actualiza el predictor BTB con dicha información.

```
void actualizarBHT (unsigned long long pc, char tomado)
```

Recibe el PC de una instrucción de salto y un booleano que indica si el salto ha sido tomado o no. Actualiza el predictor BHT con dicha información.

En un procesador real, la actualización del predictor se realiza en la etapa de consolidación. De esta forma, el predictor sólo se actualiza con la información de los saltos finalmente ejecutados. En nuestro simulador, hemos insertado las llamadas a `actualizarBTB()` y `actualizarBHT()` en la misma etapa de búsqueda, inmediatamente después de las llamadas de consulta. El simulador nos permite conocer toda la información necesaria para la actualización del predictor ya en la etapa de búsqueda.

2. Modelado de los predictores

En este apartado vas a implementar las funciones para consultar y actualizar los predictores. Para la predicción de sentido (tomado/no tomado) vamos a implementar tres

predictores: local, global, híbrido. En todos los casos, BHT tiene 2 bits por entrada. Usaremos el primer diagrama de estados de dos bits presentado en clase.

Las funciones `leerBHT() y actualizarBHT()` son genéricas. Como ejemplo, `leerBHT`, en función del valor de la variable *predictor* (valores l, g, h) llama a `leerBHTl, leerBHTg o leerBHTh` que implementan el predictor local, global o híbrido respectivamente.

Las estructuras de datos necesarias se muestran a continuación, y están disponibles en el fichero saltos.c:

```
#define logBTBsize 10
#define logBHTsize 15

typedef struct BTBentry {
    unsigned long long tag;
    unsigned long long direccion;
} BTBentry_t;

BTBentry_t BTB[1 << logBTBsize];

char BHT[1 << logBHTsize];
```

3. Estudiar comportamiento con distintos patrones

Los programas matrizc y matrizf, que ya conocéis, implementan recorridos de matrices con bucles que incorporan saltos con un patrón muy definido y fácil de predecir. Diseñad otro programa de prueba con saltos que sigan distintos patrones. Ejecutad los programas de prueba con nuestro simulador y analizad el comportamiento del predictor y su influencia en las prestaciones del procesador.